UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF VIRGINIA
ALEXANDRIA DIVISION

CELLEBRITE MOBILE SYNCHRONIZATION LTD.,
and CELLEBRITE USA, INC.

Plaintiff.

v.

MICRO SYSTEMATION AB, and
MSAB, INC.

Defendants.

Civil Action No. _____-cv-_____

EXPERT DECLARATION OF
ROBERT ZEIDMAN

ECF Case

Table of Contents

I, Robert Zeidman, provide the following expert disclosures.

I. SUMMARY OF FINDINGS

1. I conducted an in-depth comparative analysis of binary files and source code for the Universal Forensic Extraction Device ("UFED") developed by Cellebrite Mobile Synchronization Ltd. ("Cellebrite") and binary files from the XRY software application ("XRY") of Micro Systemation AB and MSAB, Inc. ("MSAB"). The UFED and XRY include forensic software that extracts data from a variety of devices. The UFED and XRY include Samsung extractions targeting devices produced by Samsung Electronics and BlackBerry extractions targeting devices produced by Research In Motion ("RIM"). As I explain below, I found Cellebrite's techniques, parameters, and even the name "Cellebrite" within MSAB's files. This is one of the most extensive collections of similarities, within binary files, that I have observed throughout my career as an expert in numerous litigations involving software. As a result of my analysis, it is my professional opinion that MSAB created their Samsung and BlackBerry solutions by directly copying techniques and expressions from Cellebrite.

II. BACKGROUND

2. This introductory section of my report gives information about my qualifications as well as a discussion of technical terms needed to understand this report.

A. PERSONAL EXPERIENCE AND BACKGROUND OF ROBERT ZEIDMAN

3. I am an engineer and the founder and president of Zeidman Consulting, which provides engineering consulting services to high-tech companies. Among the types of services I provide are hardware and software design. My clients have included Fortune 500 computer and technology companies as well as smaller companies and startups. A copy of my resume is attached hereto as Exhibit A.

4. I hold a Master's degree from Stanford University in Electrical Engineering and two

Bachelor's degrees from Cornell University, one in Electrical Engineering and one in Physics.

5. I have been a computer software and hardware designer for over 25 years. I have designed and developed a variety of computer hardware and software products. These software products include Internet-based training courses and web-based course administration software, an operating system synthesis tool, a source code comparison tool, a network emulation software bridge, and a remote backup system. I have founded several companies including eVault, a remote backup company, the Chalkboard Network, an e-learning company, Zeidman Technologies, a company that develops software tools for enabling and improving hardware and software development, and Software Analysis and Forensic Engineering Corporation, a company that develops software analysis tools.

6. I have written a variety of papers, books, and presentations on computer hardware and software and other engineering subjects. My three books on chip design are *Verilog Designer's Library*, *Introduction to Verilog*, and *Designing with FPGAs and CPLDs*. I am the developer of the Universal Design Methodology, a process for efficiently developing reliable systems, about which I have written extensively. A list of my publications is included in my resume attached as Exhibit A.

7. I hold a number of patents for software synthesis, hardware emulation, hardware synthesis, hardware simulation, and software code comparison. I have created a tool called CodeSuite® that incorporates BitMatch®, CodeCross®, CodeDiff®, CodeMatch®, CodeCLOC®, and SourceDetective® for detecting whether one computer program has been plagiarized from another computer program.

8. I have written a book on software intellectual property entitled *The Software IP Detective's Handbook: Measurement, Comparison, and Infringement Detection.* I have consulted on matters involving intellectual property disputes, including instances of alleged misappropriation and infringement. My work in this capacity has included, among other things, reviewing and analyzing software source code, reviewing and analyzing patents,

reverse engineering hardware and software, writing expert reports, and testifying in court.

9. I have testified at deposition and at trial in a number of cases involving software copyright infringement, trade secret theft, and patent infringement. The specific cases can be found in my resume, attached as Exhibit A.

B. SOURCE CODE

10. Computer programs can be written using complex instructions that look like English. For example, the instruction `a = b*c+2` tells the computer to take the number stored in memory and represented by variable `b`, multiply that by the number stored in memory and represented by the variable `c`, add 2 and store the result in memory represented by the variable `a`. Similarly, the statement `printf("Hello world!")` tells the computer to print the words "Hello world!" to the computer screen. These high-level, English-like instructions are called "source code." Computer programs are made up of many lines of source code and the process of writing these lines of code is called programming. Eventually these lines of source code are turned into binary code instructions that a computer understands, consisting of sequences of electronic ones and zeroes. The process of turning human-readable source code into a file containing computer instructions is called "compiling" and is performed by a special computer program called a "compiler." The resulting machine-readable code is called object or binary code.

C. BINARY CODE

11. Computer programs written in source code get changed to many bytes of binary computer code by a program called a compiler. The computer can then directly execute the functions described by the binary data. During this compilation, much of the human readable source code is lost, leaving binary code that is difficult for a human to read directly. However, some of the human-readable elements, such as strings and some identifiers, may be left within the binary code.

12. This very low level binary code program is also called “object code.” There are different types of object code files including “executable files” that have an extension “.exe” if they are to be used on a computer running the Microsoft Windows operating system. Another kind of object code file, called a “dynamic link library,” has an extension of “.dll” if they are to be used in the Microsoft Windows operating system. There are other kinds of object files and they have different file name extensions if they are to be used on computers running other operating systems.

D. HEXADECIMAL

13. Developers often view binary data and develop software using hexadecimal notation or simply "hex." Usually a prefix of `0x` or a suffix of `h` is used to denote a hex representation of a number. A group of four binary ones and zeros can represent a value from 0- 15. Hexadecimal notation represents values from 0 to 15 with the symbols 0-9 and A-F. For example, a sequence of four zeroes, in binary represented as `0000` is represented in hex as 0x0. Similarly a sequence of four ones, in binary represented as `1111`, is represented in hex as 0xF. For developers, the hexadecimal representation is easier to read than the binary ones and zeroes.

E. DECOMPILERS

14. Decompilers are software tools that convert binary back to source code. The decompiled source code produced by a decompiler may be functionally equivalent to the original source code. However, there will be literal differences since the original compilation usually strips out many of the human-readable elements of the source code, such as comments, and reorganizes the code. The process of compilation and decompilation is similar to translating a book from a first language into a second language, and then translating the book back from the second language back to the first language. Although, the storyline would remain intact, the translation to and from the second language would alter the exact

words and sentences and sequences of words and sentences within the book.

F. SOURCE CODE CORRELATION

15. Source code correlation is a measure of the similarity of one program's source code to another program's source code. For the purpose of measuring source code correlation, source code can be considered to consist of three kinds of elements that can be categorized as statements from which can be derived a control structure, comments that serve to document the code, and strings that are messages to users, as shown in Table 1. Statements further consist of instructions and identifiers. Instructions comprise control words and operators. Identifiers comprise variables, constants, functions, and labels. A single line of source code may include one or more statements and one or more comments and one or more strings.

Software source code elements	**Description**
Statements	Cause actions to occur. They are sequence dependent.
Instructions	Signify the actions to take place.
Control words	Control the program flow (e.g. `if`, `case`, goto, `loop`).
Operators	Manipulate data (e.g. +, -. *. /).
Identifiers	Reference code or data.
Variables	Identify data.
Constants	Identify constants.
Functions	Identify code.
Labels	Identify locations in the program.
Comments	For documentation purposes only. Cause no actions to occur.
Strings	Messages to the user.

Table 1. Software source code elements

16. The overall correlation between two source code files is measured as a combination of the correlation of several different elements of the source code: statement correlation, comment/string correlation, identifier correlation, and instruction sequence correlation.

17. I am the originator of source code correlation, and the technique has gained acceptance in engineering and in law. I first presented the detailed concept in a session entitled "Software Source Code Correlation" at the 5th IEEE/ACIS International Conference on Computer and Information Science, a peer-reviewed conference given on July 12, 2006. I also gave presentations at the Thirty First Asilomar Microcomputer Workshop, Silicon Valley Code Camp, the High Technology Crime Investigation Association, and at the peer-reviewed Third International Workshop on Systematic Approaches to Digital Forensic Engineering. I have published several articles on the topic in several magazines and journals including the law magazine Intellectual Property Today. I have given lectures on the topic at the Stanford Law School and the NALSAR University of Law in Hyderabad, India and at various other places. I have written the first book on software forensics entitled *The Software IP Detective's Handbook: Measurement, Comparison, and Infringement Detection*. These papers and presentations about source code correlation that can be found in my resume, attached as Exhibit A.

G. BINARY CODE CORRELATION

18. When source code is not available, binary files can be compared to find correlation. The binary files may include strings, comments, and identifier names. If uncommon strings, comments, and identifier names are found in two binary files, there is a strong possibility that one was copied from the other. If such correlation is found, it implies that a section of the binary file or the source code that was compiled into the binary files was copied. Likewise, if sections of two binary files are correlated, then one is likely to have been copied from the other.

H. BITMATCH

19. BitMatch® compares thousands of executable binary files in multiple directories and subdirectories to thousands of other executable binary files or source code files in order to

determine which files are the most highly correlated. BitMatch is particularly useful for finding programs that have been copied, but there is no access to the program executable binary files and not the source code.

20. BitMatch compares every file in one directory with every file in another directory, including all subdirectories if requested. BitMatch produces a database that can then be exported to an HTML basic report that lists the most highly correlated pairs of files.

21. BitMatch examines all text strings, comments, and identifier names that it can find in the executable files in order to determine copying. If a specific user message or a unique subroutine name is found in two files, there is a possibility that one was copied from the other. Note that BitMatch gives only a rough determination whether copying took place. False negatives are very possible. CodeMatch, which compares source code for correlation, is needed to compare source code to make a definitive determination.

I. BOOTLOADER

22. Electronic devices often include a bootloader, which is commonly known as a small piece of software code that starts when a device is turned on. The bootloader typically initializes the memory and peripherals, such as a screen and keyboard, and then starts an operating system running on the device. The bootloader may also verify that the software running on the device was supplied by the vendor of the device. This verification may "lock" the device to prevent alternative software from running on the device and the unauthorized extraction of user data from the device.

J. DATA EXTRACTION

23. Running unauthorized software and performing unauthorized extraction of user data can be difficult on locked devices. In order to extract user data, highly skilled developers try to find ways to circumvent the security of locked devices. This often involves circumventing the verification performed by a bootloader. Circumventing a bootloader can be a

painstaking process of finding unintended loopholes, or "vulnerabilities," in the security of a device. Device manufactures strive to prevent such vulnerabilities. If a developer finds a vulnerability, the developer can then attempt to exploit the vulnerability to run custom software, which may be incorporated in a bootloader. The custom software may extract the user data from the device. The entire process of finding a vulnerability and then developing an exploit of the vulnerability can be a very long and difficult process, involving much trial and error.

III. SCOPE OF REPORT

24. Based on my background and experience, I have been asked by the law firm of Pearl Cohen Zedek Latzer Baratz LLP, on behalf of Cellebrite Mobile Synchronization Ltd. ("Cellebrite"), to provide my opinions and conclusions related to whether all or parts of the software used by Micro Systemation AB and MSAB, Inc. ("MSAB") were copied from software created by Cellebrite.

25. In reaching the opinions and conclusions discussed herein, I received, considered, and/or relied upon the following materials:

a. Cellebrite source and binary code

b. Cellebrite configuration file `__Dump_SamsungGSM.xml`

c. Cellebrite Samsung extraction payload `QC_Exploit_Generic_from_1_1_9_0.unpacked.bin`

d. Cellebrite source code file `headers.h`

e. Cellebrite source code file `switch.h`

f. Cellebrite source code file `cmd_driver_nand.c`

g. Cellebrite source code file `cmd_alloc.c`

h. Cellebrite source code file `Patcher.h`

i. Cellebrite source code file `ClientBlackBerry-3778.py`

j. Cellebrite source code file `flash.c`

k. Cellebrite BlackBerry extraction payload `BB.bin`

l. Cellebrite patched bootloaders and exploitation payloads captured by Cellebrite developers from USB communications

m. Files extracted and captured from MSAB XRY 6.4 by Cellebrite

n. Declaration of Guy Biron in support of plaintiffs' motion for a temporary restraining order and preliminary injunctive relief

o. Declaration of Nadav Horesh in support of plaintiffs' motion for a temporary restraining order and preliminary injunctive relief

p. Samsung serial data interface document "CDMA Cellular Subscriber Station Serial Data Interface Control Document Version 1.0"

q. XRY Forensic Pack 6.5.0 software installed from XRY_FORENSIC_PACK_32bit_6.5.0.zip

r. XRY Forensic Pack 6.6.1 software installed from XRY_FORENSIC_PACK_32bit_6.6.1.zip

s. MSAB's SGH-G800 Configuration file extracted from XRY Forensic Pack 6.5.0 software

t. MSAB binary file `QCDumper.dll`

u. MSAB's Samsung extraction payload file `samsung_qcom_loader.bin`

v. MSAB binary file `BlackBerryDumper.dll`

w. MSAB BlackBerry extraction payload `xry_loader`

x. MSAB patched bootloaders and exploitation payloads captured by Cellebrite developers from USB communications

y. The BitMatch tool from Software Analysis and Forensic Engineering Corporation

z. The Interactive Disassembler (IDA) from Hex-Rays SA

IV. ANALYSIS

26. My analysis found evidence that MSAB copied both techniques and expressions from Cellebrite's Samsung and BlackBerry solutions. I have based my analysis on files that I

extracted from the MSAB's XRY Forensic pack 6.5.0. In addition, I have reviewed files and USB communications that Cellebrite developers extracted and captured from MSAB XRY 6.4 and their own Cellebrite UFED.

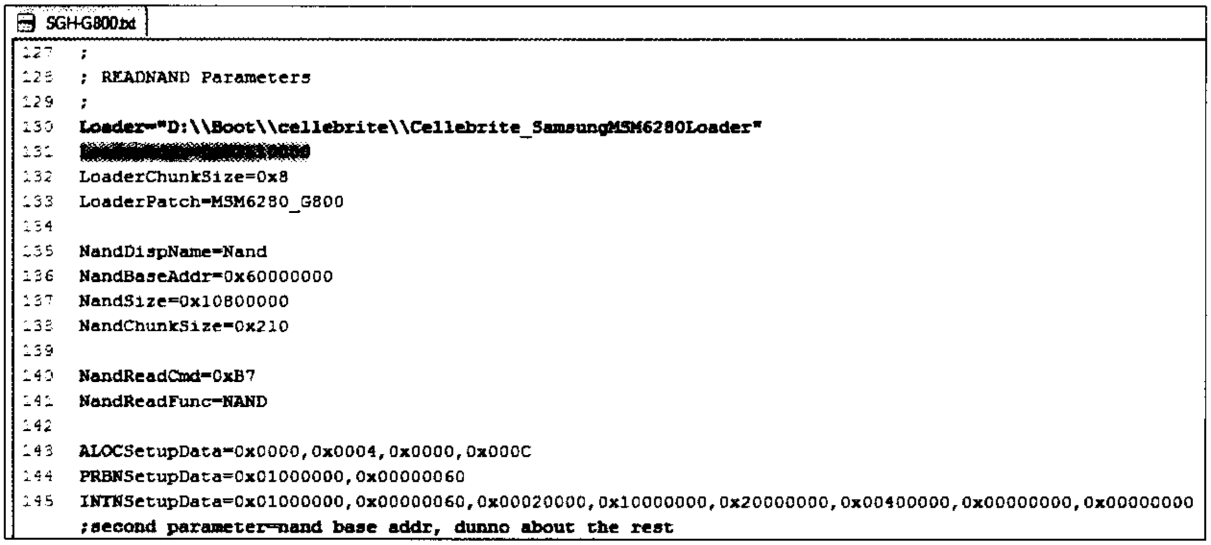
A. SAMSUNG SOLUTION

27. MSAB copied Cellebrite's Samsung extraction technique, code, and parameters. As discussed below, this copying is found in the MSAB XRY configuration files and binary files. A configuration file found within the MSAB XRY software includes the name "Cellebrite," which demonstrates that MSAB developed their Samsung solution based upon Cellebrite's software. This and other MSAB files show that MSAB directly copied several parameter values from Cellebrite's Samsung solution. The numerous similarities between the MSAB and Cellebrite Samsung solutions show that MSAB copied Cellebrite's techniques and specific expressions to create their Samsung solution.

1. MSAB CONFIGURATION FILE INCLUDES THE TERM "CELLEBRITE"

28. The extracted MSAB XRY files include a directory of device configuration files. Each of these device configuration files is labeled with a given device name and includes information and parameters regarding that device. The parameters within each device configuration file appear to specify how the MSAB XRY software communicates and extracts information from the device. When made available, an analysis of the MSAB XRY source code will likely confirm that the MSAB XRY software relies upon these configuration files.

29. The file `SGH-G800.txt` is one such configuration file; it includes parameters for performing a data extraction from a Samsung SGH-G800 smartphone. This SGH-G800 configuration file, shown in Figure 1, includes several references to Cellebrite's Samsung solution including Cellebrite parameters, Cellebrite commands, and even the name "Cellebrite."



```
127  ;
128  ; READNAND Parameters
129  ;
130  Loader="D:\\Boot\\cellebrite\\Cellebrite_SamsungMSM6280Loader"
131  [redacted]
132  LoaderChunkSize=0x8
133  LoaderPatch=MSM6280_G800
134
135  NandDispName=Nand
136  NandBaseAddr=0x60000000
137  NandSize=0x10800000
138  NandChunkSize=0x210
139
140  NandReadCmd=0xB7
141  NandReadFunc=NAND
142
143  ALOCSetupData=0x0000,0x0004,0x0000,0x000C
144  PRBNSetupData=0x01000000,0x00000060
145  INTNSetupData=0x01000000,0x00000060,0x00020000,0x10000000,0x20000000,0x00400000,0x00000000,0x00000000
     ;second parameter=nand base addr, dunno about the rest
```

Figure 1. MSAB's SGH-G800 Configuration file

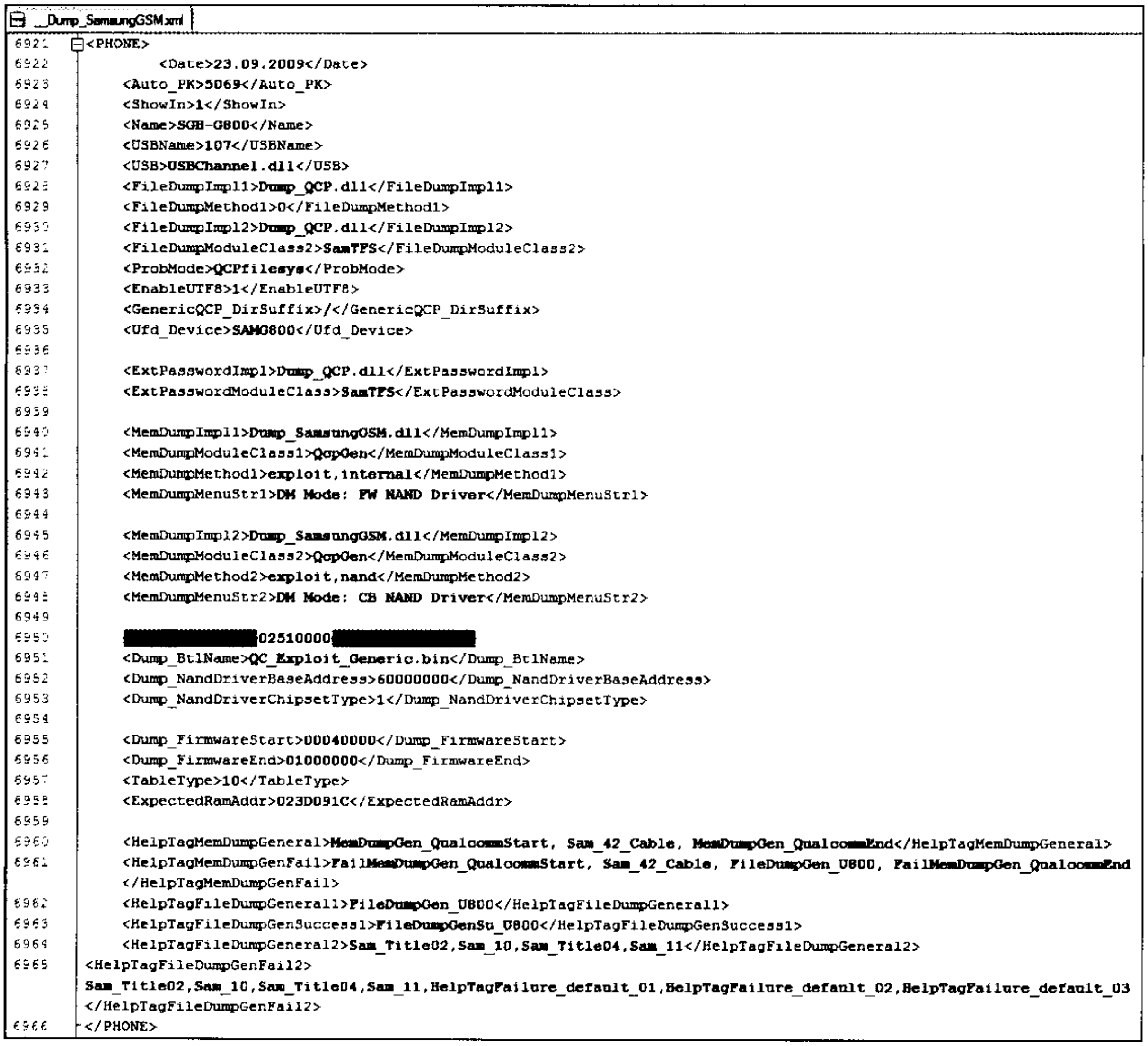
30. Line 130 of MSAB's SGH-G800 configuration file, `SGH-G800.txt`, includes the name Cellebrite. The name Cellebrite is found once in the file name `Cellebrite_SamsungMSM6289Loader` and again in the directory name `cellebrite`. This file name indicates that MSAB relied upon a Cellebrite bootloader to develop their Samsung Extractor. The name of the directory also indicates that MSAB may have relied upon additional Cellebrite material not listed in their SGH-G800 configuration file.

2. PAYLOAD UPLOAD ADDRESSES

31. The reliance upon Cellebrite material is also evident in the parameter values and command names found in the rest of MSAB's SGH-G800 configuration file. Line 131 of MSAB's SGH-G800 configuration file sets a parameter `LoaderAddr` to the value `0x02510000`. This value is the same memory address into which Cellebrite loads its extraction payload during the Cellebrite Samsung extraction. The extraction payload is a small piece of software that performs the actual data extraction. Typically, a piece of software, such as the extraction payload, can be loaded into a variety of memory addresses. For example, the memory could be thought of as a large group of chairs, where each chair is a memory

address and various pieces of software, such as an extraction payload, could be seated.

32. Developers may arbitrarily select a specific memory address, such as 0x02510000, from many potential memory addresses. Cellebrite lists configuration information for Samsung phones in a single configuration file __Dump_SamsungGSM.xml, shown in Figure 2. Specifically, __Dump_SamsungGSM.xml includes configuration information for the Samsung SGH-G800. At line 6950 of __Dump_SamsungGSM.xml the parameter Dump_BtlStart is assigned the memory address 0x02510000 for the SGH-G800. This configuration information shows that Cellebrite loads their extraction payload into the memory address 0x02510000 for the SGH-G800.



```
<PHONE>
        <Date>23.09.2009</Date>
    <Auto_PK>5069</Auto_PK>
    <ShowIn>1</ShowIn>
    <Name>SGH-G800</Name>
    <USBName>107</USBName>
    <USB>USBChannel.dll</USB>
    <FileDumpImpl1>Dump_QCP.dll</FileDumpImpl1>
    <FileDumpMethod1>0</FileDumpMethod1>
    <FileDumpImpl2>Dump_QCP.dll</FileDumpImpl2>
    <FileDumpModuleClass2>SamTFS</FileDumpModuleClass2>
    <ProbMode>QCPfilesys</ProbMode>
    <EnableUTF8>1</EnableUTF8>
    <GenericQCP_DirSuffix>/</GenericQCP_DirSuffix>
    <Ufd_Device>SAMG800</Ufd_Device>

    <ExtPasswordImpl>Dump_QCP.dll</ExtPasswordImpl>
    <ExtPasswordModuleClass>SamTFS</ExtPasswordModuleClass>

    <MemDumpImpl1>Dump_SamsungGSM.dll</MemDumpImpl1>
    <MemDumpModuleClass1>QcpGen</MemDumpModuleClass1>
    <MemDumpMethod1>exploit,internal</MemDumpMethod1>
    <MemDumpMenuStr1>DM Mode: FW NAND Driver</MemDumpMenuStr1>

    <MemDumpImpl2>Dump_SamsungGSM.dll</MemDumpImpl2>
    <MemDumpModuleClass2>QcpGen</MemDumpModuleClass2>
    <MemDumpMethod2>exploit,nand</MemDumpMethod2>
    <MemDumpMenuStr2>DM Mode: CB NAND Driver</MemDumpMenuStr2>

    [redacted]02510000[redacted]
    <Dump_BtlName>QC_Exploit_Generic.bin</Dump_BtlName>
    <Dump_NandDriverBaseAddress>60000000</Dump_NandDriverBaseAddress>
    <Dump_NandDriverChipsetType>1</Dump_NandDriverChipsetType>

    <Dump_FirmwareStart>00040000</Dump_FirmwareStart>
    <Dump_FirmwareEnd>01000000</Dump_FirmwareEnd>
    <TableType>10</TableType>
    <ExpectedRamAddr>023D091C</ExpectedRamAddr>

    <HelpTagMemDumpGeneral>MemDumpGen_QualcommStart, Sam_42_Cable, MemDumpGen_QualcommEnd</HelpTagMemDumpGeneral>
    <HelpTagMemDumpGenFail>FailMemDumpGen_QualcommStart, Sam_42_Cable, FileDumpGen_U800, FailMemDumpGen_QualcommEnd
    </HelpTagMemDumpGenFail>
    <HelpTagFileDumpGeneral1>FileDumpGen_U800</HelpTagFileDumpGeneral1>
    <HelpTagFileDumpGenSuccess1>FileDumpGenSu_U800</HelpTagFileDumpGenSuccess1>
    <HelpTagFileDumpGeneral2>Sam_Title02,Sam_10,Sam_Title04,Sam_11</HelpTagFileDumpGeneral2>
<HelpTagFileDumpGenFail2>
Sam_Title02,Sam_10,Sam_Title04,Sam_11,HelpTagFailure_default_01,HelpTagFailure_default_02,HelpTagFailure_default_03
</HelpTagFileDumpGenFail2>
</PHONE>
```

Figure 2. Cellebrite configuration data for Samsung device __Dump_SamsungGSM.xml

33. In discussion with Cellebrite developers, they confirmed that they selected the memory address for uploading the `QC_Exploit_Generic.bin` extraction payload from many possible memory addresses by testing arbitrarily memory addresses. If two parties independently select memory addresses for uploading software, then it is typically unlikely that the two parties arbitrarily select the same memory address. For example, if you attend a movie one night and your neighbor independently attends a movie the next night, it is unlikely that you would both arbitrarily select the same chair in the movie theatre.

Likewise, it is unlikely that MSAB independently decided to "seat" their extraction payload in the same memory address that Cellebrite selected. It is more likely that MSAB copied the memory address from Cellebrite rather, than independently selecting the same memory address.

34. At line 6951 of __Dump_SamsungGSM.xml, the parameter Dump_BtlName is assigned the file name QC_Exploit_Generic.bin. This file QC_Exploit_Generic.bin is the extraction payload for the Cellebrite Samsung solution. When I analyzed the configuration information of the various Samsung phones set to use the QC_Exploit_Generic.bin extraction payload, I found that Cellebrite uploads the QC_Exploit_Generic.bin extraction payload to one of five different memory addresses. The five different memory addresses are 0x02510000, 0x0141A000, 0x01800000, 0x0252F000, and 0x02800000, as shown in Exhibit B. In a discussion with Cellebrite developers, they confirmed that they selected these five memory addresses from many potential memory addresses.

35. The MSAB software includes configuration files for several Samsung devices. I searched every MSAB configuration file and found that the parameter LoaderAddr was set for 35 Samsung devices. I found that MSAB did not only select a Cellebrite memory address for the SGH-G800, but in 31 of the 35 configuration files MSAB selected the exact same memory address as Cellebrite. Furthermore, for 34 out of these 35 configuration files, MSAB set LoaderAddr to one of the five memory addresses that Cellebrite uses. It is extremely unlikely that MSAB independently selected the same memory addresses. This can be seen in Table 2, derived from files \DeviceInfo\Samsung\*.txt and ..\Genesis 1190RC 9JAN2012\Genesis\XML DB\DataFiles\Phones\__Dump_SamsungGSM.xml. The most likely explanation for the many identical selections is that MSAB directly copied from Cellebrite.

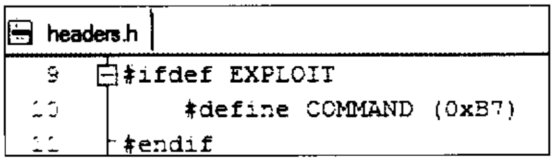
Samsung Device Name	MSAB LoaderAddr	Cellebrite Dump BitStart	Same Address	One of Cellebrite's Addresses
GT-M7500 Emporio Armani	02510000	02510000	TRUE	TRUE
GT-S7220 Ultra b	02510000	02510000	TRUE	TRUE
GT-S7350 Ultra s	02510000	02510000	TRUE	TRUE
GT-S8300 Ultra Touch	02510000	02510000	TRUE	TRUE
SGH-A597 Eternity 2	02510000	02510000	TRUE	TRUE
SGH-A637	02510000	02510000	TRUE	TRUE
SGH-A707 Sync	02510000	02510000	TRUE	TRUE
SGH-A717	02510000	02510000	TRUE	TRUE
SGH-A727	02510000	02510000	TRUE	TRUE
SGH-A737	02510000	02510000	TRUE	TRUE
SGH-A747 SLM	02510000	02510000	TRUE	TRUE
SGH-A777	02510000	02510000	TRUE	TRUE
SGH-A827 Access	02510000	02510000	TRUE	TRUE
SGH-A837 Rugby	02510000	02510000	TRUE	TRUE
SGH-A867 Eternity	02800000	02800000	TRUE	TRUE
SGH-A886 Forever	02800000	02800000	TRUE	TRUE
SGH-A887 Solstice	02510000	02510000	TRUE	TRUE
SGH-A897 Mythic	02800000	02800000	TRUE	TRUE
SGH-A927 Flight II	02800000	02800000	TRUE	TRUE
SGH-F480 Touchwiz	02510000	0252F000	FALSE	TRUE
SGH-F480v Touchwiz	01EF0000	01800000	FALSE	FALSE
SGH-F488i Touchwiz Anycall	02510000	01800000	FALSE	TRUE
SGH-F490	02510000	02510000	TRUE	TRUE
SGH-F490v	02510000	02510000	TRUE	TRUE
SGH-G800	02510000	02510000	TRUE	TRUE
SGH-L170	02510000	02510000	TRUE	TRUE
SGH-L760	02510000	02510000	TRUE	TRUE
SGH-T469 Gravity 2	02800000	02800000	TRUE	TRUE
SGH-T639	02510000	02510000	TRUE	TRUE
SGH-T746 Impact	02800000	02800000	TRUE	TRUE
SGH-T919 Behold	02800000	02800000	TRUE	TRUE
SGH-T929 Memoir	02510000	02510000	TRUE	TRUE
SGH-U800 Soul b	01800000	01800000	TRUE	TRUE
SGH-U900 Soul	02510000	02510000	TRUE	TRUE
SGH-Z560	0141A000	01800000	FALSE	TRUE

Table 2. MSAB's use of Cellebrites bootloader addresses from \XACT 6.4.1

3. OVERIDE "MAGIC" COMMAND CODE

36. Returning to the parameters in MSAB's SGH-G800 configuration file. Line 140 includes the value `0xB7` for the parameter `NandReadCmd`. An extraction may be referred to as a NAND memory read, where NAND flash memory is a common type of memory used in mobile devices. Therefore, it appears that MSAB used the `NandReadCmd` value `0xB7` to communicate with the extraction payload. A review of the MSAB XRY source code will likely confirm that the MSAB XRY uses the value `0xB7` to communicate with the extraction payload.

37. Cellebrite uses this same value `0xB7`as a command prefix for communicating with their extraction payload via Samsung diagnostic commands. Cellebrite developers call this their "magic" command. This command prefix is set in the Cellebrite source code file `..\Qualcomm code revision 1051\Shared\headers.h`, shown in Figure 3.



```
#ifdef EXPLOIT
    #define COMMAND (0xB7)
#endif
```

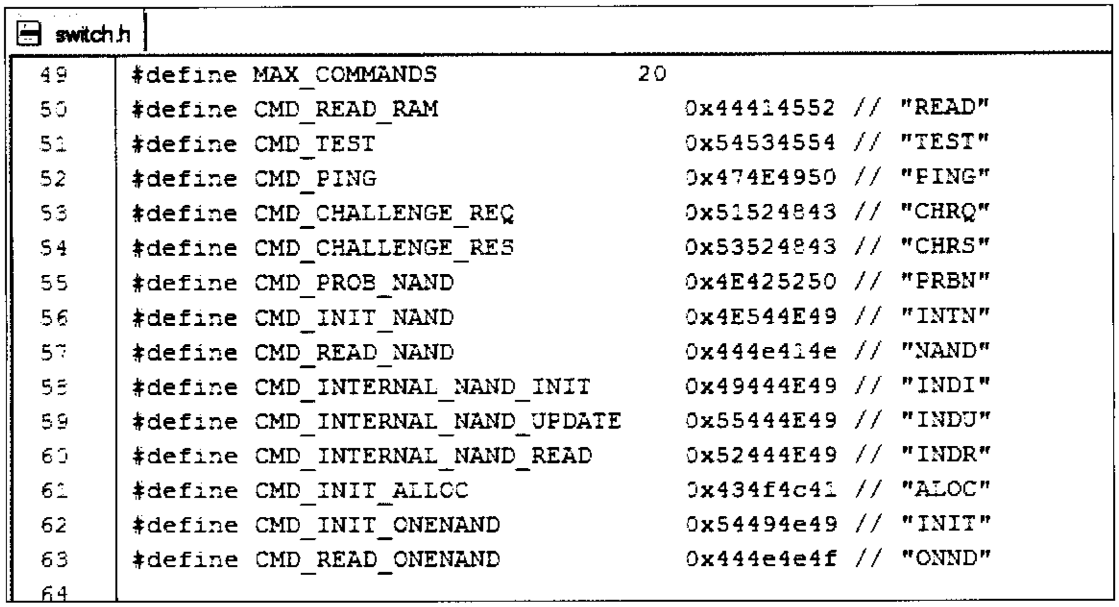
Figure 3. Cellebrite exploit command 0xB7 from ..\Qualcomm code revision 1051\Shared\headers.h

38. There are many possible Samsung diagnostic commands that Cellebrite could have used for their solution, but according to Cellebrite developers, they arbitrarily chose `0xB7` from about 180 available Samsung diagnostic command codes. The serial data interface of the Samsung CDMA Cellular Dual Mode Subscriber System (DMSS) identifies Samsung diagnostic commands with an 8 bit command code. Eight bits allows for 256 unique Samsung diagnostic command codes. Exhibit C is a table from Samsung's Serial Data Interface Control Document that lists the supported Samsung diagnostic commands. As shown in the exhibit, Samsung has assigned command codes 0x00 - 0x69 for 70 Samsung diagnostic commands. This leaves over 180 available command codes, hexadecimal 0x70 - 0xFF, from which Cellebrite arbitrarily chose `0xB7`.

39. As shown, MSAB's configuration file includes the same value, `0xB7`, that Cellebrite had arbitrarily selected. As before, the most likely explanation for this identical value, `0xB7`, is that MSAB directly copied Cellebrite's technique and expression.

4. CELLEBRITE'S COMMANDS IN MSAB XRY

40. The Cellebrite source code file `..\Qualcomm code revision 1051\Shared\switch.h`, shown in Figure 4, defines several commands in the Cellebrite source code. These commands include the `NAND`, `ALOC`, `PRBN`, and `INTN` at lines 57, 61, 55, and 56 respectively.



switch.h

```
49   #define MAX_COMMANDS                    20
50   #define CMD_READ_RAM                       0x44414552 // "READ"
51   #define CMD_TEST                           0x54534554 // "TEST"
52   #define CMD_PING                           0x474E4950 // "PING"
53   #define CMD_CHALLENGE_REQ                  0x51524843 // "CHRQ"
54   #define CMD_CHALLENGE_RES                  0x53524843 // "CHRS"
55   #define CMD_PROB_NAND                      0x4E425250 // "PRBN"
56   #define CMD_INIT_NAND                      0x4E544E49 // "INTN"
57   #define CMD_READ_NAND                      0x444e414e // "NAND"
58   #define CMD_INTERNAL_NAND_INIT             0x49444E49 // "INDI"
59   #define CMD_INTERNAL_NAND_UPDATE           0x55444E49 // "INDU"
60   #define CMD_INTERNAL_NAND_READ             0x52444E49 // "INDR"
61   #define CMD_INIT_ALLOC                     0x434f4c41 // "ALOC"
62   #define CMD_INIT_ONENAND                   0x54494e49 // "INIT"
63   #define CMD_READ_ONENAND                   0x444e4e4f // "ONND"
64
```

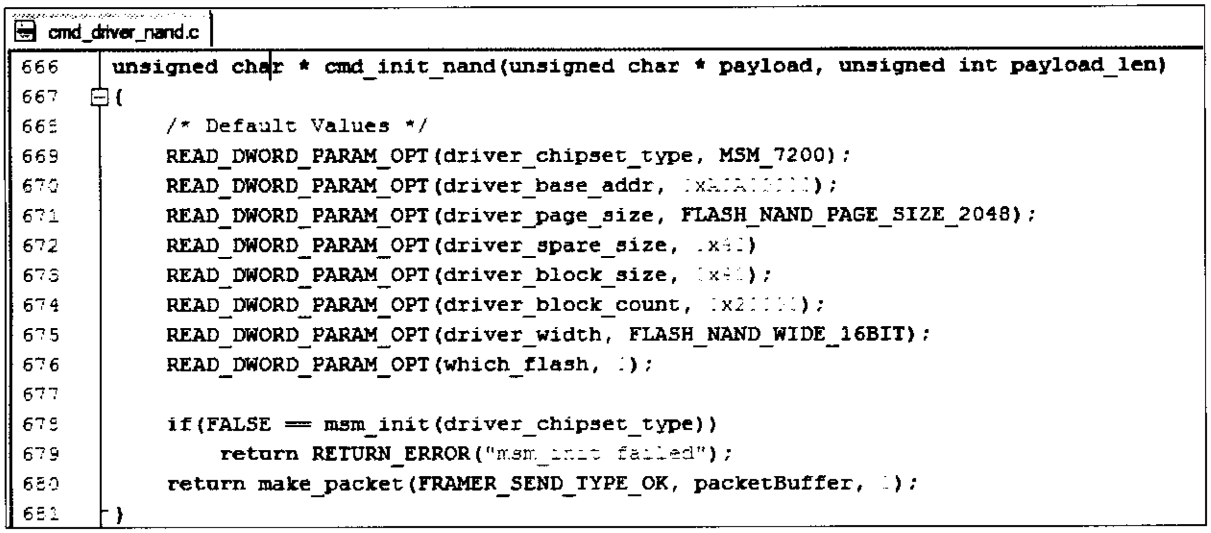
Figure 4. Cellebrite bootloader commands from ..\Qualcomm code revision 1051\Shared\switch.h

41. As shown in Figure 1, the lines 141, 143, 144, and 145 of MSAB's SGH-G800 configuration file include the strings: `NAND`, `ALOC`, `PRBN`, and `INTN`. Not only do these strings match Cellebrite's commands, but the parameters following the strings correspond to the parameters that Cellebrite uses for these commands.

42. For instance, line 56 of the Cellebrite file `switch.h` defines the Cellebrite command `INTN` as `CMD_INIT_NAND`. Cellebrite's corresponding function `cmd_init_nand` is

"

implemented in `..\Qualcomm code revision 1051\Shared\cmd\cmd_driver_nand.c`, Figure 5. This Cellebrite function `cmd_init_nand` sets eight parameters. As shown in Figure 1, the setting `INTNSetupData` at line 145 of the MSAB SGH-G800 configuration file includes the same number of parameters, eight. The matching strings and parameters are most likely the result of MSAB copying Cellebrite's commands.



```
unsigned char * cmd_init_nand(unsigned char * payload, unsigned int payload_len)
{
    /* Default Values */
    READ_DWORD_PARAM_OPT(driver_chipset_type, MSM_7200);
    READ_DWORD_PARAM_OPT(driver_base_addr, 0xA0A00000);
    READ_DWORD_PARAM_OPT(driver_page_size, FLASH_NAND_PAGE_SIZE_2048);
    READ_DWORD_PARAM_OPT(driver_spare_size, 0x40)
    READ_DWORD_PARAM_OPT(driver_block_size, 0x40);
    READ_DWORD_PARAM_OPT(driver_block_count, 0x20000);
    READ_DWORD_PARAM_OPT(driver_width, FLASH_NAND_WIDE_16BIT);
    READ_DWORD_PARAM_OPT(which_flash, 0);

    if(FALSE == msm_init(driver_chipset_type))
        return RETURN_ERROR("msm_init failed");
    return make_packet(FRAMER_SEND_TYPE_OK, packetBuffer, 0);
}
```

Figure 5. Cellebrite command cmd_init_nand with 8 parameters from ..\Qualcomm code revision 1051\Shared\cmd\cmd_driver_nand.c

43. As shown in Figure 1, line 145 of MSAB's SGH-G800 configuration file also includes a comment "`second parameter=nand base addr, dunno about the rest.`" Apparently, whoever wrote this comment only understood the purpose of one of the eight parameters. It would be very unusual for a developer not to understand code he or she had written. However, it would not be surprising for a developer not to understand software he or she had copied. Therefore, this MSAB comment is possibly an artifact of MSAB developers failing to understand software that they copied from Cellebrite.

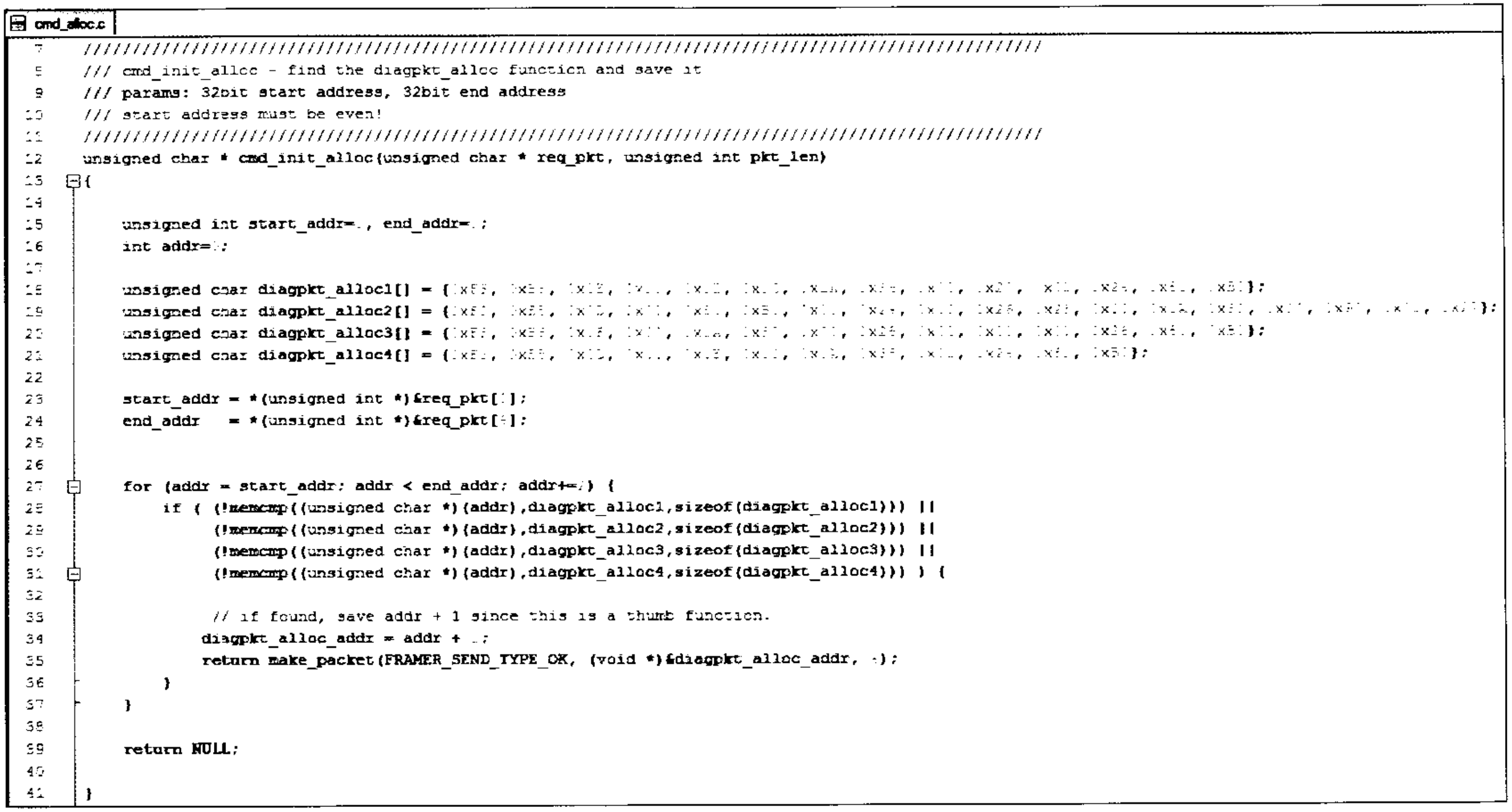
44. Furthermore, when I ran a BitMatch comparison between the Cellebrite source code and MSAB binary files, I found three of these Cellebrite commands, `NAND`, `PRBN`, and `INTN`, within the MSAB binary code file `QCDumper.dll`. The BitMatch results are included as

Exhibit D. Thus, these commands were not just part of MSAB's SGH-G800 configuration file, but they are also included within MSAB's binary code. The most likely explanation for the inclusion of the Cellebrite commands in MSAB's `QCDumper.dll` is that MSAB copied the commands from Cellebrite.

5. USB FUNCTION SIGNATURES

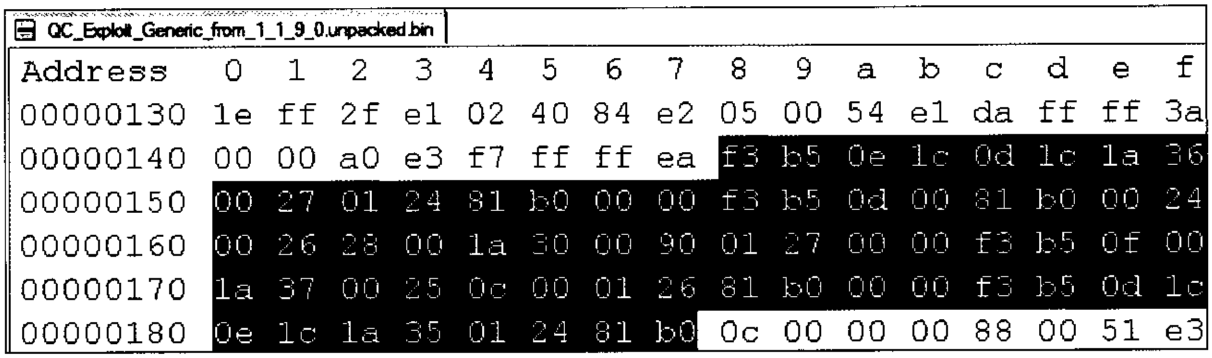
45. The Samsung solution communicates with Cellebrite's UFED via a universal serial bus (USB) connection. There are a number of ways to setup this USB communication. The Cellebrite extraction payload sets up the USB communication by locating the Samsung USB functions within the Samsung software. The various Samsung devices that Cellebrite supports include at least one of four different Samsung USB functions. To identify the Samsung USB function used in a specific device, Cellebrite developed a unique method of searching for "signatures" of the Samsung USB functions. As part of developing this method, Cellebrite developers collected four such signatures. Each signature matches some portion of a Samsung USB function. During a Samsung extraction, the Cellebrite extraction payload may have to perform multiple searches before one of the Cellebrite signatures matches the particular Samsung USB function used in the device.

46. The Cellebrite function `cmd_init_alloc` starting at line 7 of the source code file `..\Samsung\Qualcomm code revision 1051\Exploit\Files\cmd_alloc.c`, shown in Figure 6, identifies the USB functions within a Samsung device. There are many ways to search within a section of software, so this function `cmd_init_alloc` could have been implemented in a variety of ways. The four signatures collected by Cellebrite are set at lines 18 to 21 of the source code file `cmd_alloc.c`. The four signatures are also found within a hexadecimal representation of Cellebrite's Samsung extraction payload `QC_Exploit_Generic_from_1_1_9_0.unpacked.bin`, shown in Figure 7.



```
////////////////////////////////////////////////////////////////////////////////////////////////////////
/// cmd_init_alloc - find the diagpkt_alloc function and save it
/// params: 32bit start address, 32bit end address
/// start address must be even!
////////////////////////////////////////////////////////////////////////////////////////////////////////
unsigned char * cmd_init_alloc(unsigned char * req_pkt, unsigned int pkt_len)
{

    unsigned int start_addr=[illegible], end_addr=[illegible];
    int addr=[illegible];

    unsigned char diagpkt_alloc1[] = {[illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible]};
    unsigned char diagpkt_alloc2[] = {[illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible]};
    unsigned char diagpkt_alloc3[] = {[illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible]};
    unsigned char diagpkt_alloc4[] = {[illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible], [illegible]};

    start_addr = *(unsigned int *)&req_pkt[[illegible]];
    end_addr   = *(unsigned int *)&req_pkt[[illegible]];


    for (addr = start_addr; addr < end_addr; addr+=[illegible]) {
        if ( (!memcmp((unsigned char *)(addr),diagpkt_alloc1,sizeof(diagpkt_alloc1))) ||
             (!memcmp((unsigned char *)(addr),diagpkt_alloc2,sizeof(diagpkt_alloc2))) ||
             (!memcmp((unsigned char *)(addr),diagpkt_alloc3,sizeof(diagpkt_alloc3))) ||
             (!memcmp((unsigned char *)(addr),diagpkt_alloc4,sizeof(diagpkt_alloc4))) ) {

             // if found, save addr + 1 since this is a thumb function.
            diagpkt_alloc_addr = addr + [illegible];
            return make_packet(FRAMER_SEND_TYPE_OK, (void *)&diagpkt_alloc_addr, [illegible]);
        }
    }

    return NULL;

}
```

Figure 6. Cellebrite signatures from ..\Samsung\Qualcomm code revision 1051\Exploit\Files\cmd_alloc.c

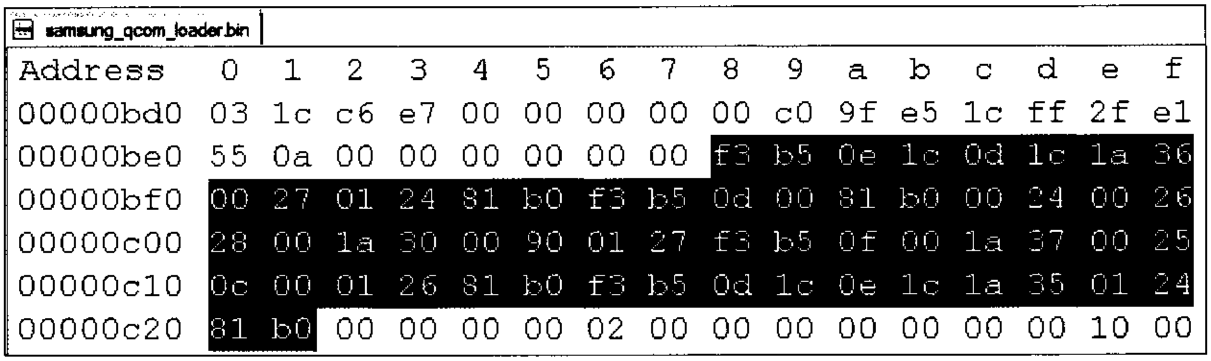


QC_Exploit_Generic_from_1_1_9_0.unpacked.bin

```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000130 1e ff 2f e1 02 40 84 e2 05 00 54 e1 da ff ff 3a
00000140 00 00 a0 e3 f7 ff ff ea f3 b5 0e 1c 0d 1c 1a 36
00000150 00 27 01 24 81 b0 00 00 f3 b5 0d 00 81 b0 00 24
00000160 00 26 28 00 1a 30 00 90 01 27 00 00 f3 b5 0f 00
00000170 1a 37 00 25 0c 00 01 26 81 b0 00 00 f3 b5 0d 1c
00000180 0e 1c 1a 35 01 24 81 b0 0c 00 00 00 88 00 51 e3
```

Figure 7. Cellebrite's signatures within Cellebrite Samsung extraction payload QC_Exploit_Generic_from_1_1_9_0.unpacked.bin

47. Although the content of the signatures is dependent on the Samsung USB functions, Cellebrite's choice of the specific signature is arbitrary. If two developers were independently choosing such signatures, it is unlikely that the two developers would create signatures with the exact same length that match the exact same portions of the Samsung USB functions. However, MSAB's Samsung extraction payload file `samsung_qcom_loader.bin`, shown in Figure 8, includes the exact same signatures. It is unlikely that MSAB independently developed the exact same technique for finding the Samsung USB functions and even more unlikely that MSAB independently selected the exact same signatures.



samsung_qcom_loader.bin

```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000bd0 03 1c c6 e7 00 00 00 00 00 c0 9f e5 1c ff 2f e1
00000be0 55 0a 00 00 00 00 00 00 f3 b5 0e 1c 0d 1c 1a 36
00000bf0 00 27 01 24 81 b0 f3 b5 0d 00 81 b0 00 24 00 26
00000c00 28 00 1a 30 00 90 01 27 f3 b5 0f 00 1a 37 00 25
00000c10 0c 00 01 26 81 b0 f3 b5 0d 1c 0e 1c 1a 35 01 24
00000c20 81 b0 00 00 00 00 02 00 00 00 00 00 00 00 10 00
```

Figure 8. Cellebrite's signatures within MSAB Samsung extraction payload samsung_qcom_loader.bin

B. BLACKBERRY SOLUTION

48. I also analyzed MSAB's BlackBerry solution and found that MSAB copied Cellebrite's BlackBerry solution. I also the USB communication data that the Cellebrite developers captured from the Cellebrite and MSAB BlackBerry extractions. The captured data appears to represent the USB communications from the Cellebrite and MSAB BlackBerry extractions accurately, and includes further indications that MSAB copied from Cellebrite. The numerous similarities between the MSAB and Cellebrite BlackBerry solutions show that MSAB's copied Cellebrite's techniques and expressions to create their BlackBerry solution.

1. SAME VERSION OF BOOTLOADER

49. Cellebrite's BlackBerry solution utilizes BlackBerry signed bootloaders. According to Cellebrite developers, they developed a complicated process to extract the BlackBerry signed bootloaders from a version of the BlackBerry desktop software that they downloaded in April 2011. I also understand that MSAB released their BlackBerry solution in November 2012, over one year after Cellebrite extracted the BlackBerry signed bootloaders.

50. I reviewed archived versions of the BlackBerry desktop software page using the "Wayback Machine" of the archive.org website that tracks web pages. The pages I examined included, for example, `http://web.archive.org/web/20120624012005/http://us.blackberry.com/software/desktop.html`. I found that RIM updated the BlackBerry desktop software on their website from version 6.0 to version 7.0 in June 2012 and then from version 7.0 to version 7.1 in August 2012. It is likely that RIM also produced minor updates in-between these major updates.

51. Considering the BlackBerry desktop software update schedule, if two companies independently extracted the BlackBerry bootloaders from the BlackBerry desktop software,

then it is unlikely that they would extract the same version. With multiple versions of the BlackBerry bootloaders available between April 2011 and November 2012, one would expect MSAB to use a later version of the BlackBerry bootloaders. However, when I compared the binary content of the 29 bootloaders that MSAB uses, shown in Table 3, I found that all 29 were identical to the BlackBerry bootloaders that Cellebrite extracted in April 2011. The most likely explanation for MSAB using the same BlackBerry bootloaders that Cellebrite painstakingly extracted is that MSAB copied the BlackBerry bootloaders from Cellebrite's BlackBerry solution.

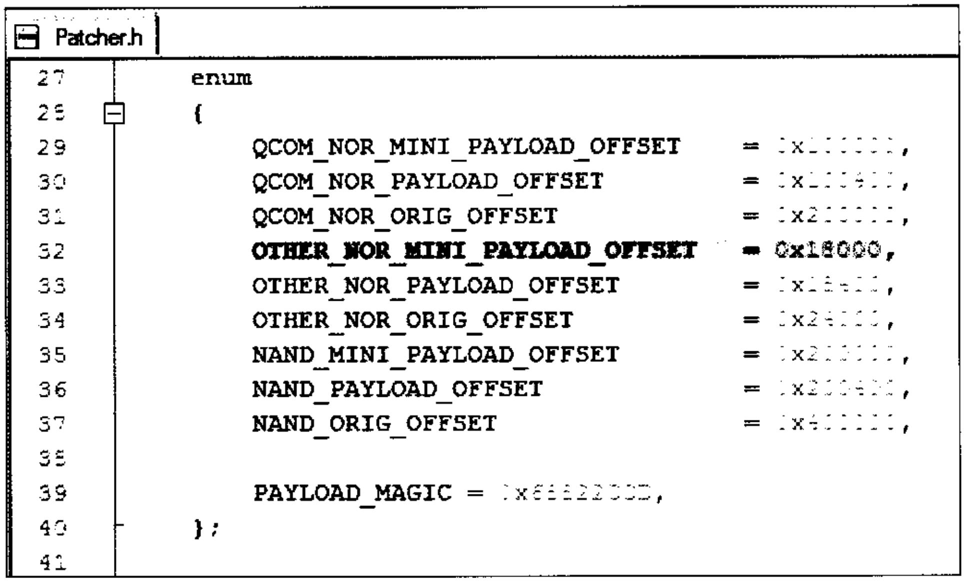
MSAB Bootloaders	Cellebrite Bootloaders	Identical File Contents
blackberry_8110_loader	8F000D03_0000000A	TRUE
blackberry_8120_loader	8D000D03_0000000A	TRUE
blackberry_8130_loader	06000D04_0000000A	TRUE
blackberry_8220_loader	8D001103_0000000A	TRUE
blackberry_8230_loader	06001104_0000000A	TRUE
blackberry_8300_loader	96000F03_0000000A	TRUE
blackberry_8310_loader	8D000F03_0000000A	TRUE
blackberry_8320_loader	84000F03_0000000A	TRUE
blackberry_8330_loader	06000F04_0000000A	TRUE
blackberry_8350i_loader	84000F05_0000000A	TRUE
blackberry_8520_loader	85000F03_0000000A	TRUE
blackberry_8800_loader	84000E03_0000000A	TRUE
blackberry_8900_loader	84001503_0000000A	TRUE
blackberry_8910_loader	86001503_0000000A	TRUE
blackberry_9000_loader	84000E07_0000000A	TRUE
blackberry_9100_loader	05000D07_0000000A	TRUE
blackberry_9105_loader	05000D07_0000000A	TRUE
blackberry_9500_loader	07001404_0000000A	TRUE
blackberry_9550_loader	0E001404_0000000A	TRUE
blackberry_9630_loader	16000D04_0000000A	TRUE
blackberry_9700_loader	04001507_0000000A	TRUE
blackberry_9780_loader	15001507_0000000A	TRUE
blackberry_9800_loader	05001807_0000000A	TRUE
blackberry_9810_loader	0C001804_0000000A	TRUE
blackberry_9850_loader	16001404_0000000A	TRUE
blackberry_9860_loader	1D001404_0000000A	TRUE
blackberry_9900_loader	07001204_0000000A	TRUE
blackberry_9930_loader	05001204_0000000A	TRUE

Table 3. Comparison of Cellebrite and MSAB extracted bootloaders

2. LOADING ADDRESS

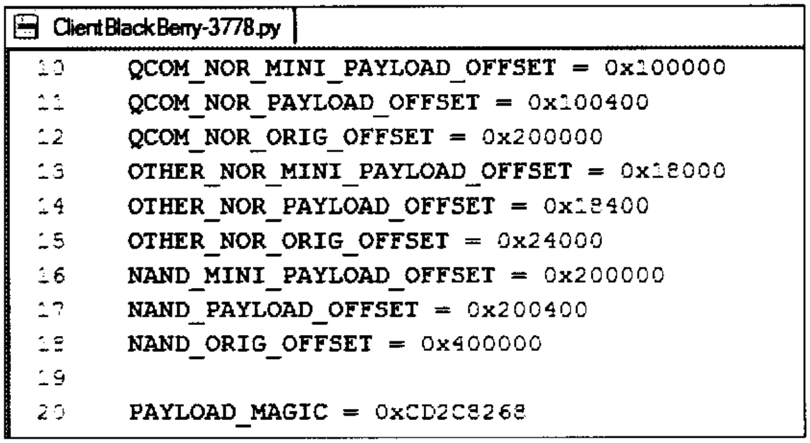
52. The Cellebrite BlackBerry solution includes uploading the original BlackBerry bootloader, a patched bootloader, and an extraction payload that is coupled with a stack changer function. In Cellebrite's BlackBerry solution, only the patched bootloader has to be uploaded to a specific memory address to work with the BlackBerry smartphone. The other two pieces of software are uploaded to arbitrarily chosen memory addresses. The memory

addresses that Cellebrite arbitrarily chose are found in the Cellebrite source code file `Patcher.h`, shown in Figure 9, and the Cellebrite source code file `ClientBlackBerry-3778.py`, shown in Figure 10.



```
        enum
        {
            QCOM_NOR_MINI_PAYLOAD_OFFSET     = 0x100000,
            QCOM_NOR_PAYLOAD_OFFSET          = 0x100400,
            QCOM_NOR_ORIG_OFFSET             = 0x200000,
            OTHER_NOR_MINI_PAYLOAD_OFFSET    = 0x18000,
            OTHER_NOR_PAYLOAD_OFFSET         = 0x18400,
            OTHER_NOR_ORIG_OFFSET            = 0x24000,
            NAND_MINI_PAYLOAD_OFFSET         = 0x200000,
            NAND_PAYLOAD_OFFSET              = 0x200400,
            NAND_ORIG_OFFSET                 = 0x400000,

            PAYLOAD_MAGIC = 0x6112200D,
        };
```

Figure 9. Source code file Patcher.h



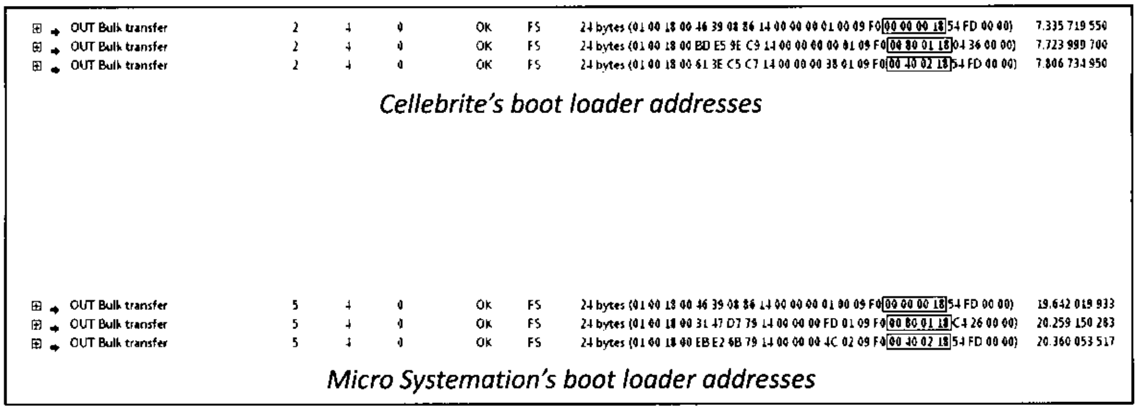
```
QCOM_NOR_MINI_PAYLOAD_OFFSET = 0x100000
QCOM_NOR_PAYLOAD_OFFSET = 0x100400
QCOM_NOR_ORIG_OFFSET = 0x200000
OTHER_NOR_MINI_PAYLOAD_OFFSET = 0x18000
OTHER_NOR_PAYLOAD_OFFSET = 0x18400
OTHER_NOR_ORIG_OFFSET = 0x24000
NAND_MINI_PAYLOAD_OFFSET = 0x200000
NAND_PAYLOAD_OFFSET = 0x200400
NAND_ORIG_OFFSET = 0x400000

PAYLOAD_MAGIC = 0xCD2C8268
```

Figure 10. Source coide file ClientBlackBerry-3778.py

53. As discussed above, developers may arbitrarily select to upload to a specific memory address from many potential memory addresses. Sometimes, developers specify memory address offsets between various pieces of software, instead of absolute memory addresses.

Cellebrite used different memory address offsets for different types of BlackBerry devices. One set of memory address offsets includes the values `0x18000`, `0x18400`, and `0x24000`, as seen at lines 32-34 of Cellebrite source code file `Patcher.h`, or lines 13-15 of Cellebrite source code file `ClientBlackBerry-3778.py`.

54. I reviewed USB communications data that the Cellebrite developers captured from the Cellebrite UFED during a BlackBerry extraction, shown in Figure 11. From my discussions with the Cellebrite developers and review of the data, the data appear to represent the USB communications from the Cellebrite UFED accurately. The Cellebrite USB communications include the same address offsets `0x18000`, `0x18400`, and `0x24000`, as the Cellebrite source code. Note that the USB communications encode the memory address offsets in little endian notation, which reads right to left. This encoding is common in computer science.



```
⊞ ➔ OUT Bulk transfer    2    4    0    OK    FS    24 bytes (01 00 18 00 46 39 08 86 14 00 00 00 01 00 09 F0 00 00 00 18 54 FD 00 00)    7.335 719 550
⊞ ➔ OUT Bulk transfer    2    4    0    OK    FS    24 bytes (01 00 18 00 BD E5 9E C9 14 00 00 00 00 01 09 F0 00 80 01 18 04 36 00 00)    7.723 990 700
⊞ ➔ OUT Bulk transfer    2    4    0    OK    FS    24 bytes (01 00 18 00 61 3E C5 C7 14 00 00 00 38 01 09 F0 00 40 02 18 54 FD 00 00)    7.806 734 950
```

Cellebrite's boot loader addresses

```
⊞ ➔ OUT Bulk transfer    5    4    0    OK    FS    24 bytes (01 00 18 00 46 39 08 86 14 00 00 00 01 00 09 F0 00 00 00 18 54 FD 00 00)    19.642 019 933
⊞ ➔ OUT Bulk transfer    5    4    0    OK    FS    24 bytes (01 00 18 00 31 47 D7 79 14 00 00 00 FD 01 09 F0 00 80 01 18 C4 26 00 00)    20.259 150 283
⊞ ➔ OUT Bulk transfer    5    4    0    OK    FS    24 bytes (01 00 18 00 EB E2 6B 79 14 00 00 00 4C 02 09 F0 00 40 02 18 54 FD 00 00)    20.360 053 517
```

Micro Systemation's boot loader addresses

Figure 11. Memory address offsets found in Cellebrite and MSAB usb communications

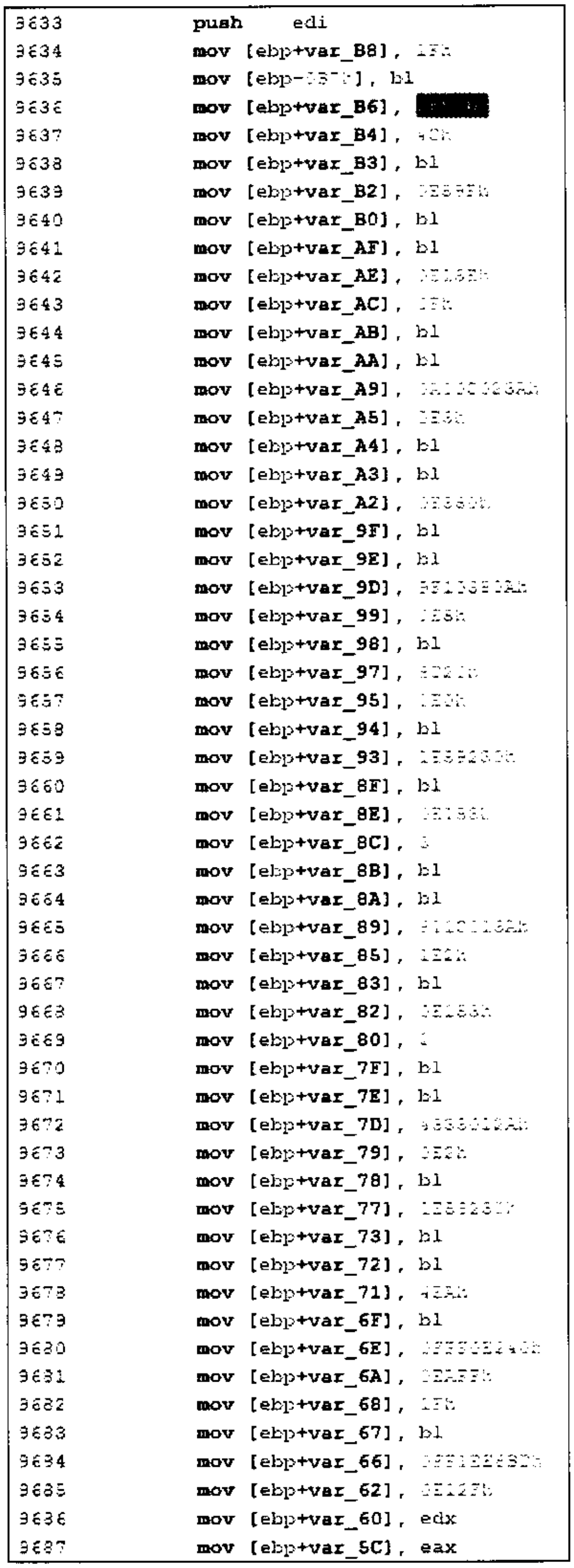
55. I also reviewed USB communications data that the Cellebrite developers captured during a MSAB BlackBerry extraction. From my discussions with the Cellebrite developers and review of the data, the data appear to represent the USB communications from the MSAB BlackBerry extraction accurately. The MSAB USB communications include the same address offsets `0x18000`, `0x18400`, and `0x24000`, as the Cellebrite USB communications and the Cellebrite source code. Again, it is extremely unlikely that two developers would

independently select the same address offsets, so the most likely explanation for the identical address offsets is that MSAB directly copied the addresses from Cellebrite.

3. STACK CHANGER FUNCTION

56. The Cellebrite BlackBerry solution includes a small software program called the stack changer function. As discussed in Guy Biron's declaration, this stack changer function forces the BlackBerry processor that is executing the BlackBerry bootloader to start executing the Cellebrite bootloader. I found the stack changer function in the Cellebrite source code file of `ClientBlackBerry-3778.py`. The Cellebrite script file `ClientBlackBerry-3778.py` inserts the entire stack changer function and three parameters into the Cellebrite extraction payload. The three parameters inserted with the stack changer function are, `orig_addr`, `orig_offset`, and `main`. The `orig_addr` parameter is the original address that the signed BlackBerry bootloader would be loaded into during normal operation. The `orig_offset` parameter is the memory address offset that the signed BlackBerry bootloader is loaded into during a Cellebrite BlackBerry extraction. The `main` parameter is the return address within the signed BlackBerry bootloader that the stack changer function modifies. The values for these parameters vary for different BlackBerry devices with different signed bootloaders.

57. Upon decompiling the MSAB binary file `..\Forensic Pack\WizardModules\Dumpers\6.4\BlackBerryDumper.dll` with the IDA disassembler, I found a sequence of binary data that is virtually identical to the Cellebrite stack changer function, as shown in Figure 12. This sequence of assembly commands loads binary data that is nearly identical to Cellebrite's stack changer function into an MSAB bootloader. The sequence of assembly commands also loads parameters at the end of the stack changer function, similar to the Cellebrite stack changer function. Note that variable `b1` is set to zero and the binary data is show in hexadecimal format using little endian notation, which reads right to left.



```
3633            push    edi
3634            mov [ebp+var_B8], 1Fh
3635            mov [ebp-[illegible]], bl
3636            mov [ebp+var_B6], [redacted]
3637            mov [ebp+var_B4], 4Ch
3638            mov [ebp+var_B3], bl
3639            mov [ebp+var_B2], [illegible]
3640            mov [ebp+var_B0], bl
3641            mov [ebp+var_AF], bl
3642            mov [ebp+var_AE], [illegible]
3643            mov [ebp+var_AC], 1Fh
3644            mov [ebp+var_AB], bl
3645            mov [ebp+var_AA], bl
3646            mov [ebp+var_A9], [illegible]
3647            mov [ebp+var_A5], 0E6h
3648            mov [ebp+var_A4], bl
3649            mov [ebp+var_A3], bl
3650            mov [ebp+var_A2], [illegible]
3651            mov [ebp+var_9F], bl
3652            mov [ebp+var_9E], bl
3653            mov [ebp+var_9D], [illegible]
3654            mov [ebp+var_99], 0E8h
3655            mov [ebp+var_98], bl
3656            mov [ebp+var_97], [illegible]
3657            mov [ebp+var_95], 1E0h
3658            mov [ebp+var_94], bl
3659            mov [ebp+var_93], [illegible]
3660            mov [ebp+var_8F], bl
3661            mov [ebp+var_8E], [illegible]
3662            mov [ebp+var_8C], 3
3663            mov [ebp+var_8B], bl
3664            mov [ebp+var_8A], bl
3665            mov [ebp+var_89], [illegible]
3666            mov [ebp+var_85], 1E2h
3667            mov [ebp+var_83], bl
3668            mov [ebp+var_82], [illegible]
3669            mov [ebp+var_80], 1
3670            mov [ebp+var_7F], bl
3671            mov [ebp+var_7E], bl
3672            mov [ebp+var_7D], [illegible]
3673            mov [ebp+var_79], 0E2h
3674            mov [ebp+var_78], bl
3675            mov [ebp+var_77], [illegible]
3676            mov [ebp+var_73], bl
3677            mov [ebp+var_72], bl
3678            mov [ebp+var_71], 4EAh
3679            mov [ebp+var_6F], bl
3680            mov [ebp+var_6E], [illegible]
3681            mov [ebp+var_6A], 0EAFFh
3682            mov [ebp+var_68], 1Fh
3683            mov [ebp+var_67], bl
3684            mov [ebp+var_66], [illegible]
3685            mov [ebp+var_62], 0E12Fh
3686            mov [ebp+var_60], edx
3687            mov [ebp+var_5C], eax
```

Figure 12. Stack changer function in BlackBerryDumper.dll

58. A side-by-side comparison of the Cellebrite stack changer function that I found in Cellebrite's source code file `ClientBlackBerry-3778.py` and the binary content that I extracted from MSAB's binary file `BlackBerryDumper.dll` is shown in Table 4. The most likely explanation for the virtually identical binary content is that MSAB directly copied the technique and expression of Cellebrite's stack changer function.

Cellebrite stack changer	MSAB stack changer
1F 00 2D E9	1F 00 2D E9
50 00 9F E5	4C 00 9F E5
00 00 5E E1	00 00 5E E1
10 00 00 3A	0F 00 00 3A
02 0C A0 E3	02 0C A0 E3
00 00 50 E3	00 00 50 E3
0D 00 00 0A	0C 00 00 0A
[redacted] 10 [redacted]	[redacted] 10 [redacted]
[redacted] 91 [redacted]	[redacted] 20 [redacted]
[redacted] 20 [redacted]	[redacted] 92 [redacted]
02 00 53 E1	01 00 53 E1
06 00 00 3A	05 00 00 3A
01 2C 82 E2	01 1C 81 E2
02 00 53 E1	01 00 53 E1
03 00 00 2A	02 00 00 2A
1C 40 9F E5	01 35 43 E2
04 30 43 E0	
00 30 81 E5	00 30 82 E5
01 00 00 EA	01 00 00 EA
04 00 40 E2	04 00 40 E2
EF FF FF EA	F0 FF FF EA
1F 00 BD E8	1F 00 BD E8
1E FF 2F E1	1E FF 2F E1

Table 4. Comparison of Cellebrite and MSAB stack changer binary contents

59. The Cellebrite developers also provided captures of the USB communication data from Cellebrite and MSAB BlackBerry extractions, which I reviewed. These captures included the same binary content that I found within the Cellebrite file `ClientBlackBerry-`

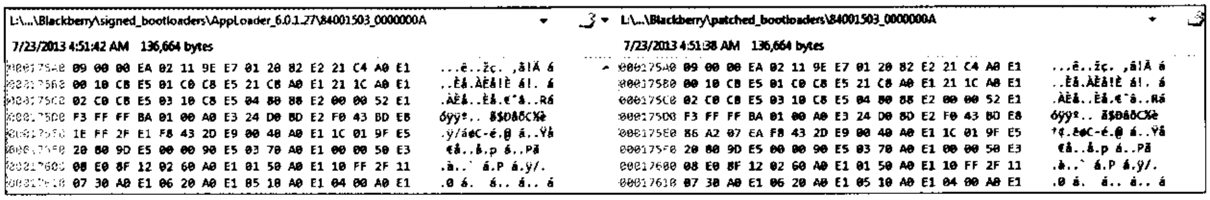
`3778.py` and the MSAB file `ClientBlackBerry-3778.py`. The captured stack changer functions also included the parameters that both Cellebrite and MSAB insert with the stack changer function. I performed a disassembly, using the IDA disassembler, of the two stack changer functions that I identified were combined with the parameters captured by the Cellebrite developers. The result of the disassembly is shown in Table 5. I have highlighted sections of the disassembly to show the similarities between the Cellebrite stack changer and the MSAB stack changer. As shown, the MSAB stack changer implements nearly identical functionality to the Cellebrite stack changer. This nearly identical functionality is most likely the result of MSAB copying from Cellebrite's BlackBerry solution.

Cellebrite disassembly	MSAB disassembly
STMFD SP!, {R0-R4}	STMFD SP!, {R0-R4}
LDR R0, =0x80400000	LDR R0, =0x80400000
CMP LR, R0	CMP LR, R0
BCC [redacted]	BCC [redacted]
MOV R0, #0x200	MOV R0, #0x200
loc_14 ; CODE XREF: ROM:00000050□j	loc_14 ; CODE XREF: ROM:0000004C□j
CMP R0, #0	CMP R0, #0
BEQ [redacted]	BEQ [redacted]
[redacted] R1, [redacted]	[redacted] R1, =0x80408CC8
[redacted] [R1]	[redacted] R2, [redacted]
[redacted] R2, =0x80408CC8	[redacted] [R2]
CMP R3, R2	CMP R3, R1
BCC loc_4C	BCC loc_48
ADD R2, R2, #0x100	ADD R1, R1, #0x100
CMP R3, R2	CMP R3, R1
BCS loc_4C	BCS loc_48
LDR R4, =0x400000	SUB R3, R3, #0x400000
SUB R3, R3, R4	STR R3, [R2]
STR R3, [R1]	B loc_50
B loc_54	
	[redacted] ; CODE XREF: ROM:0000002C□j
[redacted] ; CODE XREF: ROM:0000002C□j	; ROM:00000038□j
; ROM:00000038□j	SUB R0, R0, #4
SUB R0, R0, #4	B loc_14
B loc_14	
	[redacted] ; CODE XREF: ROM:0000000C□j
[redacted] ; CODE XREF: ROM:0000000C□j	; ROM:00000018□j ...
; ROM:00000018□j ...	LDMFD SP!, {R0-R4}
LDMFD SP!, {R0-R4}	BX LR
BX LR	
dword_5C DCD 0x80400000	dword_58 DCD 0x80400000
dword_60 DCD 0x400000	
dword_64 DCD 0x80408CC8	dword_58 DCD 0x80400000

Table 5. Comparison of disassembled Cellebrite and MSAB stack changer

4. AES ENCRYPTION FUNCTION AND COMMAND #8

60. The Cellebrite BlackBerry solution relies upon the Cellebrite patched bootloader. Cellebrite created the Cellebrite patched bootloader by replacing or "patching" two sections of the BlackBerry bootloader. I compared a BlackBerry bootloader to a Cellebrite patched bootloader, which Cellebrite developers delivered to me, and found the two patches. Figure 13 shows the first patch performed by Cellebrite. The original BlackBerry bootloader is on the left, and Cellebrite's patched bootloader is on the right. The small patched snippet is highlighted in red. According to Cellebrite developers, this patch alters the BlackBerry AES encryption function, which they arbitrarily selected from dozens of possible functions.

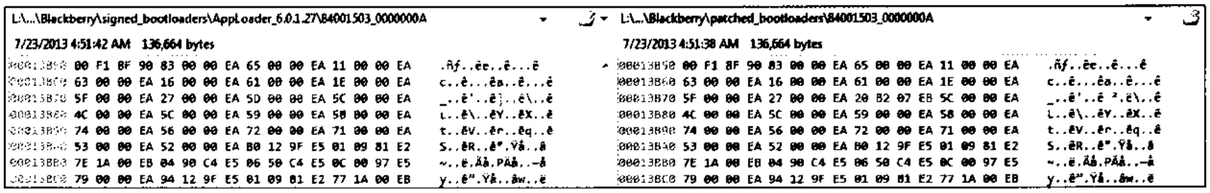


```
L:\...\Blackberry\signed_bootloaders\AppLoader_6.0.1.27\84001503_0000000A
7/23/2013 4:51:42 AM   136,664 bytes
000175A0 09 00 00 EA 02 11 9E E7 01 20 82 E2 21 C4 A0 E1   ...ê..žç. ,â!Ä á
000175B0 00 10 CB E5 01 C0 C8 E5 21 C8 A0 E1 21 1C A0 E1   ..Ëå.ÀÈå!È á!. á
000175C0 02 C0 CB E5 03 10 C8 E5 04 80 88 E2 00 00 52 E1   .ÀËå..Èå.€ˆâ..Rá
000175D0 F3 FF FF BA 01 00 A0 E3 24 D0 8D E2 F0 43 BD E8   óÿÿº.. ã$Ðâð C½è
000175E0 1E FF 2F E1 F8 43 2D E9 00 40 A0 E1 1C 01 9F E5   .ÿ/áøC-é.@ á..Ÿå
000175F0 20 80 9D E5 00 00 90 E5 03 70 A0 E1 00 00 50 E3    €å..å.p á..Pã
00017600 08 E0 8F 12 02 60 A0 E1 01 50 A0 E1 10 FF 2F 11   .à..` á.P á.ÿ/.
00017610 07 30 A0 E1 06 20 A0 E1 05 10 A0 E1 04 00 A0 E1   .0 á.  á.. á.. á
```

```
L:\...\Blackberry\patched_bootloaders\84001503_0000000A
7/23/2013 4:51:38 AM   136,664 bytes
000175A0 09 00 00 EA 02 11 9E E7 01 20 82 E2 21 C4 A0 E1   ...ê..žç. ,â!Ä á
000175B0 00 10 CB E5 01 C0 C8 E5 21 C8 A0 E1 21 1C A0 E1   ..Ëå.ÀÈå!È á!. á
000175C0 02 C0 CB E5 03 10 C8 E5 04 80 88 E2 00 00 52 E1   .ÀËå..Èå.€ˆâ..Rá
000175D0 F3 FF FF BA 01 00 A0 E3 24 D0 8D E2 F0 43 BD E8   óÿÿº.. ã$Ðâð C½è
000175E0 86 A2 07 EA F8 43 2D E9 00 40 A0 E1 1C 01 9F E5   †¢.êøC-é.@ á..Ÿå
000175F0 20 80 9D E5 00 00 90 E5 03 70 A0 E1 00 00 50 E3    €å..å.p á..Pã
00017600 08 E0 8F 12 02 60 A0 E1 01 50 A0 E1 10 FF 2F 11   .à..` á.P á.ÿ/.
00017610 07 30 A0 E1 06 20 A0 E1 05 10 A0 E1 04 00 A0 E1   .0 á.  á.. á.. á
```

Figure 13. The AES encryption function patch in Cellebrite's patched bootloader

61. Figure 14 shows the second patch performed by Cellebrite. According to Cellebrite developers, they arbitrarily selected to patch this USB communications command, command #8, selecting from among 11 such unused commands.



```
L:\...\Blackberry\signed_bootloaders\AppLoader_6.0.1.27\84001503_0000000A
7/23/2013 4:51:42 AM   136,664 bytes
00013B50 00 F1 8F 90 83 00 00 EA 65 00 00 EA 11 00 00 EA   .ñƒ..êe..ê...ê
00013B60 63 00 00 EA 16 00 00 EA 61 00 00 EA 1E 00 00 EA   c..ê...êa..ê...ê
00013B70 5F 00 00 EA 27 00 00 EA 5D 00 00 EA 5C 00 00 EA   _..ê'..ê]..ê\..ê
00013B80 4C 00 00 EA 5C 00 00 EA 59 00 00 EA 58 00 00 EA   L..ê\..êY..êX..ê
00013B90 74 00 00 EA 56 00 00 EA 72 00 00 EA 71 00 00 EA   t..êV..êr..êq..ê
00013BA0 53 00 00 EA 52 00 00 EA B0 12 9F E5 01 09 81 E2   S..êR..ê°.Ÿå..â
00013BB0 7E 1A 00 EB 04 90 C4 E5 06 50 C4 E5 0C 00 97 E5   ~..ë.Äå.PÄå..—å
00013BC0 79 00 00 EA 94 12 9F E5 01 09 81 E2 77 1A 00 EB   y..ê”.Ÿå..âw..ë
```

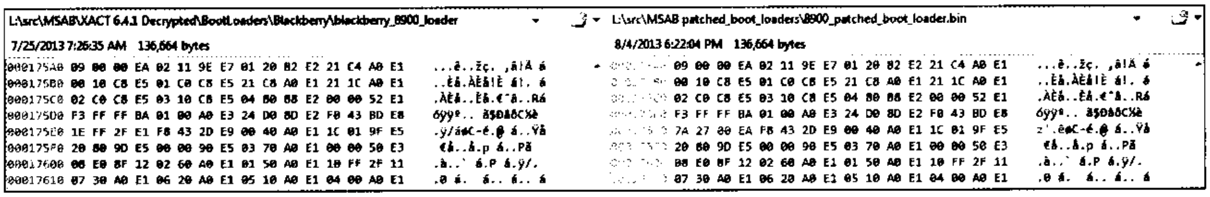
```
L:\...\Blackberry\patched_bootloaders\84001503_0000000A
7/23/2013 4:51:38 AM   136,664 bytes
00013B50 00 F1 8F 90 83 00 00 EA 65 00 00 EA 11 00 00 EA   .ñƒ..êe..ê...ê
00013B60 63 00 00 EA 16 00 00 EA 61 00 00 EA 1E 00 00 EA   c..ê...êa..ê...ê
00013B70 5F 00 00 EA 27 00 00 EA 20 B2 07 EB 5C 00 00 EA   _..ê'..ê ².ë\..ê
00013B80 4C 00 00 EA 5C 00 00 EA 59 00 00 EA 58 00 00 EA   L..ê\..êY..êX..ê
00013B90 74 00 00 EA 56 00 00 EA 72 00 00 EA 71 00 00 EA   t..êV..êr..êq..ê
00013BA0 53 00 00 EA 52 00 00 EA B0 12 9F E5 01 09 81 E2   S..êR..ê°.Ÿå..â
00013BB0 7E 1A 00 EB 04 90 C4 E5 06 50 C4 E5 0C 00 97 E5   ~..ë.Äå.PÄå..—å
00013BC0 79 00 00 EA 94 12 9F E5 01 09 81 E2 77 1A 00 EB   y..ê”.Ÿå..âw..ë
```

Figure 14. The Command #8 patch in Cellebrite's patched bootloader

62. I also reviewed MSAB patched bootloaders that were delivered to me by Cellebrite developers. I found the exact same locations patched in the MSAB patched bootloaders, showing that MSAB patched their bootloaders at the exact same locations as Cellebrite. That is, MSAB patched the BlackBerry AES encryption function, shown in Figure 15, and

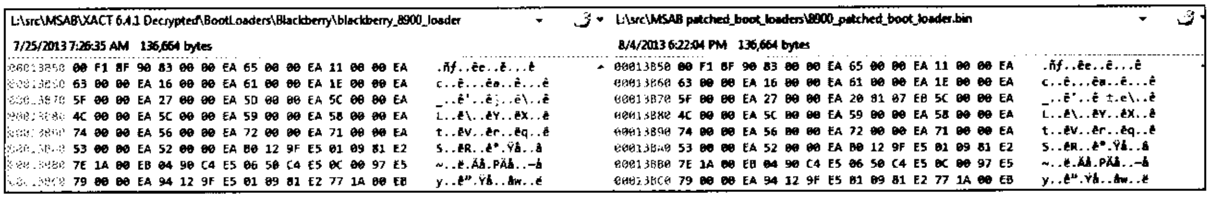
the BlackBerry command #8, shown in Figure 16.

63. The Cellebrite developers also found the use of command #8 while capturing the USB communications data of an MSAB BlackBerry solution, as shown in Figure 17. Considering there are many functions and commands that MSAB could have patched, this shows that MSAB directly copied Cellebrite's technique and expression.



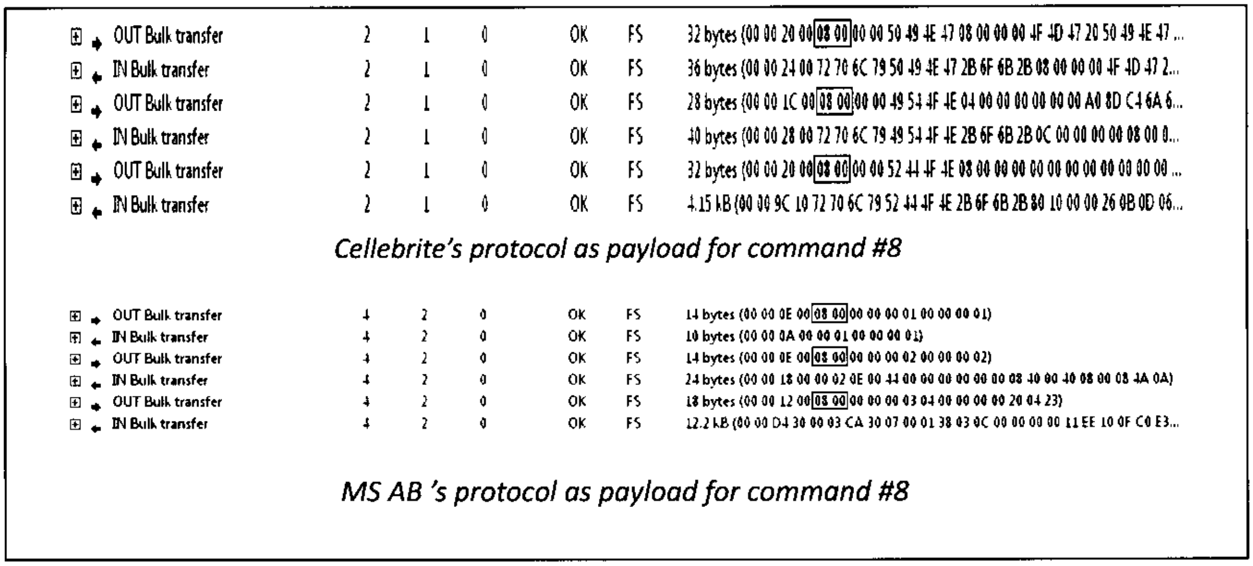
```
L:\src\MSAB\XACT 6.4.1 Decrypted\BootLoaders\Blackberry\blackberry_8900_loader
7/25/2013 7:26:35 AM   136,664 bytes
000175A0 09 00 00 EA 02 11 9E E7 01 20 82 E2 21 C4 A0 E1   ...ê..žç. ‚â!Ä á
000175B0 00 10 C8 E5 01 C0 C8 E5 21 C8 A0 E1 21 1C A0 E1   ..Èå.ÀÈå!È á!. á
000175C0 02 C0 C8 E5 03 10 C8 E5 04 80 88 E2 00 00 52 E1   .ÀÈå..Èå.€ˆâ..Rá
000175D0 F3 FF FF BA 01 00 A0 E3 24 D0 8D E2 F0 43 BD E8   óÿÿº.. ã$Ð.âðC½è
000175E0 1E FF 2F E1 F8 43 2D E9 00 40 A0 E1 1C 01 9F E5   .ÿ/áøC-é.@ á..Ÿå
000175F0 20 80 9D E5 00 00 90 E5 03 70 A0 E1 00 00 50 E3    €.å...å.p á..Pã
00017600 08 E0 8F 12 02 60 A0 E1 01 50 A0 E1 10 FF 2F 11   .à..`á.P á.ÿ/.
00017610 07 30 A0 E1 06 20 A0 E1 05 10 A0 E1 04 00 A0 E1   .0 á. á.. á.. á

L:\src\MSAB patched_boot_loaders\8900_patched_boot_loader.bin
8/4/2013 6:22:04 PM   136,664 bytes
09 00 00 EA 02 11 9E E7 01 20 82 E2 21 C4 A0 E1   ...ê..žç. ‚â!Ä á
00 10 C8 E5 01 C0 C8 E5 21 C8 A0 E1 21 1C A0 E1   ..Èå.ÀÈå!È á!. á
02 C0 C8 E5 03 10 C8 E5 04 80 88 E2 00 00 52 E1   .ÀÈå..Èå.€ˆâ..Rá
F3 FF FF BA 01 00 A0 E3 24 D0 8D E2 F0 43 BD E8   óÿÿº.. ã$Ð.âðC½è
7A 27 00 EA F8 43 2D E9 00 40 A0 E1 1C 01 9F E5   z'.êøC-é.@ á..Ÿå
20 80 9D E5 00 00 90 E5 03 70 A0 E1 00 00 50 E3    €.å...å.p á..Pã
08 E0 8F 12 02 60 A0 E1 01 50 A0 E1 10 FF 2F 11   .à..`á.P á.ÿ/.
07 30 A0 E1 06 20 A0 E1 05 10 A0 E1 04 00 A0 E1   .0 á. á.. á.. á
```

Figure 15. The AES encryption function patch in MSAB's patched bootloader



```
L:\src\MSAB\XACT 6.4.1 Decrypted\BootLoaders\Blackberry\blackberry_8900_loader
7/25/2013 7:26:35 AM   136,664 bytes
00013B50 00 F1 8F 90 83 00 00 EA 65 00 00 EA 11 00 00 EA   .ñf..êe..ê...ê
00013B60 63 00 00 EA 16 00 00 EA 61 00 00 EA 1E 00 00 EA   c..ê...êa..ê...ê
00013B70 5F 00 00 EA 27 00 00 EA 5D 00 00 EA 5C 00 00 EA   _..ê'..ê]..ê\..ê
00013B80 4C 00 00 EA 5C 00 00 EA 59 00 00 EA 58 00 00 EA   L..ê\..êY..êX..ê
00013B90 74 00 00 EA 56 00 00 EA 72 00 00 EA 71 00 00 EA   t..êV..êr..êq..ê
00013BA0 53 00 00 EA 52 00 00 EA B0 12 9F E5 01 09 81 E2   S..êR..ê°.Ÿå..â
00013BB0 7E 1A 00 EB 04 90 C4 E5 06 50 C4 E5 0C 00 97 E5   ~..ë.Äå.PÄå..—å
00013BC0 79 00 00 EA 94 12 9F E5 01 09 81 E2 77 1A 00 EB   y..ê”.Ÿå..âw..ë

L:\src\MSAB patched_boot_loaders\8900_patched_boot_loader.bin
8/4/2013 6:22:04 PM   136,664 bytes
00013B50 00 F1 8F 90 83 00 00 EA 65 00 00 EA 11 00 00 EA   .ñf..êe..ê...ê
00013B60 63 00 00 EA 16 00 00 EA 61 00 00 EA 1E 00 00 EA   c..ê...êa..ê...ê
00013B70 5F 00 00 EA 27 00 00 EA 20 81 07 EB 5C 00 00 EA   _..ê'..ê ..ë\..ê
00013B80 4C 00 00 EA 5C 00 00 EA 59 00 00 EA 58 00 00 EA   L..ê\..êY..êX..ê
00013B90 74 00 00 EA 56 00 00 EA 72 00 00 EA 71 00 00 EA   t..êV..êr..êq..ê
00013BA0 53 00 00 EA 52 00 00 EA B0 12 9F E5 01 09 81 E2   S..êR..ê°.Ÿå..â
00013BB0 7E 1A 00 EB 04 90 C4 E5 06 50 C4 E5 0C 00 97 E5   ~..ë.Äå.PÄå..—å
00013BC0 79 00 00 EA 94 12 9F E5 01 09 81 E2 77 1A 00 EB   y..ê”.Ÿå..âw..ë
```

Figure 16. The Command #8 patch in MSAB's patched bootloader



```
⊞ ➡ OUT Bulk transfer   2   1   0   OK   FS   32 bytes (00 00 20 00 08 00 00 00 50 49 4E 47 08 00 00 00 4F 4D 47 20 50 49 4E 47 ...
⊞ ⬅ IN Bulk transfer    2   1   0   OK   FS   36 bytes (00 00 24 00 72 70 6C 79 50 49 4E 47 2B 6F 6B 2B 08 00 00 00 4F 4D 47 2...
⊞ ➡ OUT Bulk transfer   2   1   0   OK   FS   28 bytes (00 00 1C 00 08 00 00 00 49 54 4F 4E 04 00 00 00 00 00 00 A0 8D C4 6A 6...
⊞ ⬅ IN Bulk transfer    2   1   0   OK   FS   40 bytes (00 00 28 00 72 70 6C 79 49 54 4F 4E 2B 6F 6B 2B 0C 00 00 00 00 08 00 0...
⊞ ➡ OUT Bulk transfer   2   1   0   OK   FS   32 bytes (00 00 20 00 08 00 00 00 52 44 4F 4E 08 00 00 00 00 00 00 00 00 00 00 00 ...
⊞ ⬅ IN Bulk transfer    2   1   0   OK   FS   4.15 kB (00 00 9C 10 72 70 6C 79 52 44 4F 4E 2B 6F 6B 2B 80 10 00 00 26 0B 0D 06...
```

Cellebrite's protocol as payload for command #8

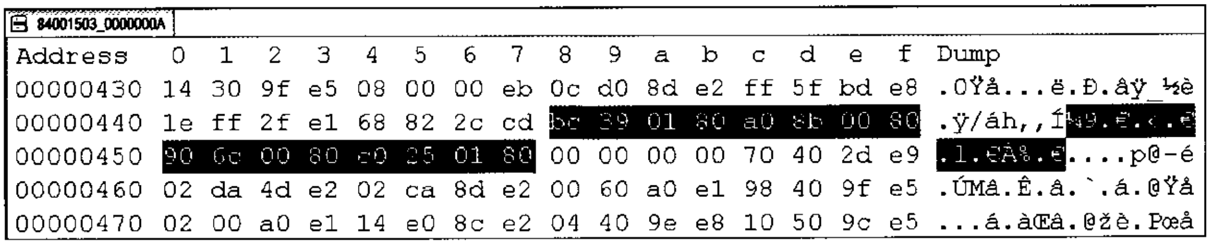
```
⊞ ➡ OUT Bulk transfer   4   2   0   OK   FS   14 bytes (00 00 0E 00 08 00 00 00 00 01 00 00 00 01)
⊞ ⬅ IN Bulk transfer    4   2   0   OK   FS   10 bytes (00 00 0A 00 00 01 00 00 00 01)
⊞ ➡ OUT Bulk transfer   4   2   0   OK   FS   14 bytes (00 00 0E 00 08 00 00 00 00 02 00 00 00 02)
⊞ ⬅ IN Bulk transfer    4   2   0   OK   FS   24 bytes (00 00 18 00 00 02 0E 00 44 00 00 00 00 00 00 08 40 00 40 08 00 08 4A 0A)
⊞ ➡ OUT Bulk transfer   4   2   0   OK   FS   18 bytes (00 00 12 00 08 00 00 00 00 03 04 00 00 00 00 20 04 23)
⊞ ⬅ IN Bulk transfer    4   2   0   OK   FS   12.2 kB (00 00 D4 30 00 03 CA 30 07 00 01 38 03 0C 00 00 00 00 11 EE 10 0F C0 E3...
```

MS AB 's protocol as payload for command #8

Figure 17. Cellebrite and MSAB use of command #8

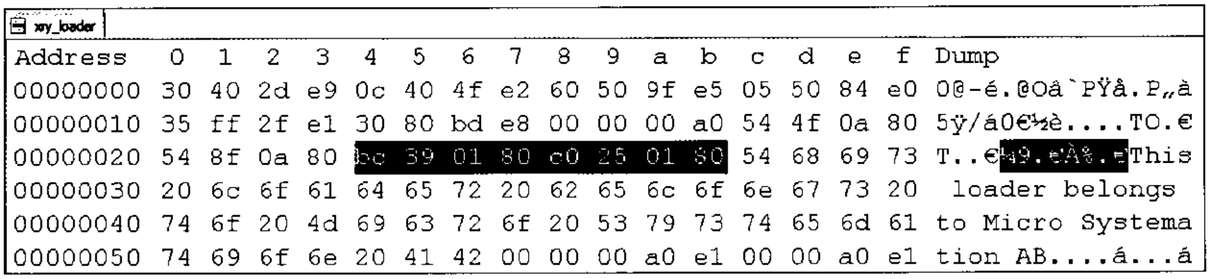
5. USB AND CACHE PARAMETERS

64. According to Cellebrite developers, the Cellebrite BlackBerry solution retrieves and sets USB and cache parameters within the Cellebrite extraction payload to enable communications. The hex parameters BC 39 01 80 and C0 25 01 80 are loaded into one such Cellebrite extraction payload that Cellebrite developers captured, shown in Figure 18. I also reviewed the MSAB extraction payload, which Cellebrite developers found in MSAB XRY 6.4 and found these same parameters, as shown in Figure 19. These matching parameters are yet another indication that MSAB copied from Cellebrite.



```
84001503_0000000A
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000430 14 30 9f e5 08 00 00 eb 0c d0 8d e2 ff 5f bd e8 .0Ÿå...ë.Ð.âÿ_½è
00000440 1e ff 2f e1 68 82 2c cd bc 39 01 80 a0 8b 00 80 .ÿ/áh‚,Í¼9.€ ‹.€
00000450 90 6c 00 80 c0 25 01 80 00 00 00 00 70 40 2d e9 .l.€À%.€....p@-é
00000460 02 da 4d e2 02 ca 8d e2 00 60 a0 e1 98 40 9f e5 .ÚMâ.Ê.â.`.á.@Ÿå
00000470 02 00 a0 e1 14 e0 8c e2 04 40 9e e8 10 50 9c e5 ...á.àŒâ.@žè.Pœå
```

Figure 18. USB and Cache parameters in Cellebrite extraction payload

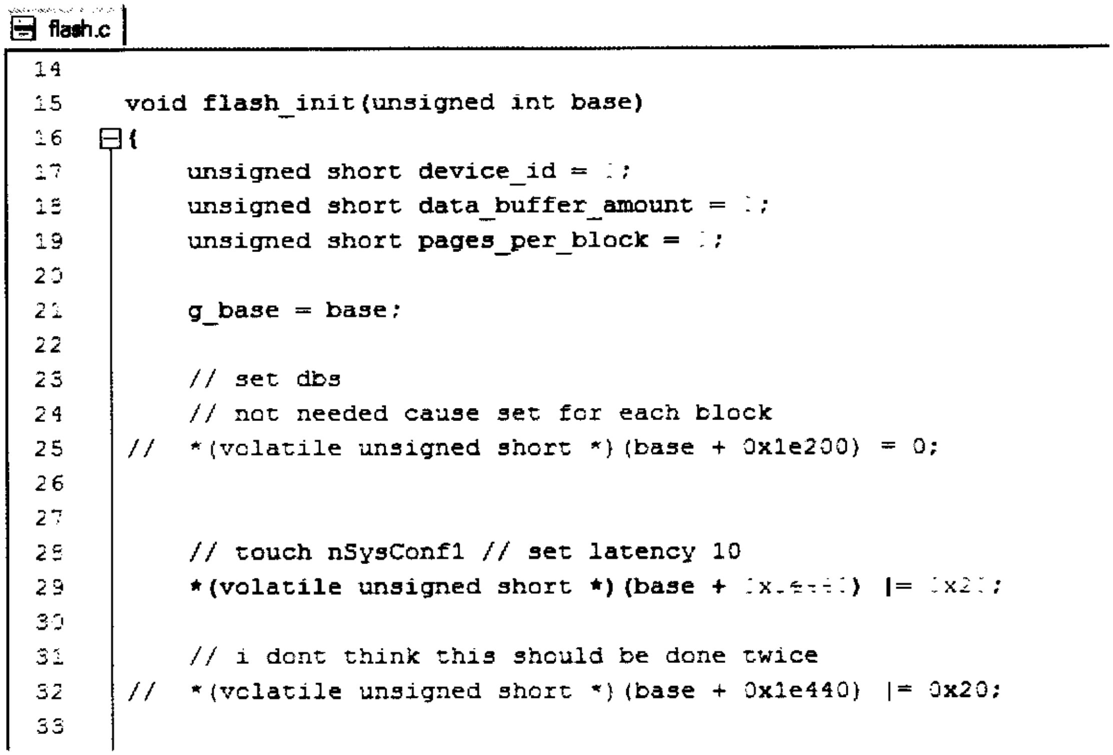


```
xry_loader
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 30 40 2d e9 0c 40 4f e2 60 50 9f e5 05 50 84 e0 0@-é.@Oâ`PŸå.P„à
00000010 35 ff 2f e1 30 80 bd e8 00 00 00 a0 54 4f 0a 80 5ÿ/á0€½è....TO.€
00000020 54 8f 0a 80 bc 39 01 80 c0 25 01 80 54 68 69 73 T..€¼9.€À%.€This
00000030 20 6c 6f 61 64 65 72 20 62 65 6c 6f 6e 67 73 20  loader belongs
00000040 74 6f 20 4d 69 63 72 6f 20 53 79 73 74 65 6d 61 to Micro Systema
00000050 74 69 6f 6e 20 41 42 00 00 00 a0 e1 00 00 a0 e1 tion AB....á...á
```

Figure 19. USB and Cache parameters in MSAB extraction payload

6. ONENAND OPERATION

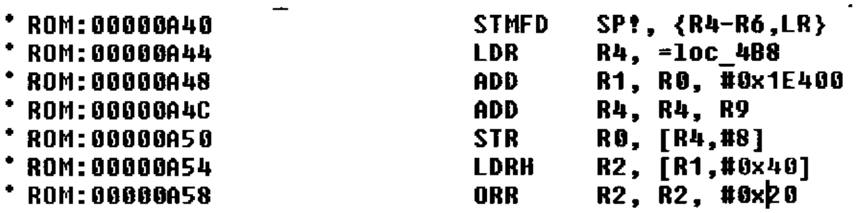
65. In order to extract data from the memory of a BlackBerry device, the Cellebrite BlackBerry solution first initializes the OneNAND memory of the device. Part of the OneNAND memory initialization is in the Cellebrite function `flash_init` in `flash.c`, shown as Figure 20.



```
14
15     void flash_init(unsigned int base)
16    {
17         unsigned short device_id = 0;
18         unsigned short data_buffer_amount = 0;
19         unsigned short pages_per_block = 0;
20
21         g_base = base;
22
23         // set dbs
24         // not needed cause set for each block
25     //  *(volatile unsigned short *)(base + 0x1e200) = 0;
26
27
28         // touch nSysConf1 // set latency 10
29         *(volatile unsigned short *)(base + 0x1e440) |= 0x20;
30
31         // i dont think this should be done twice
32     //  *(volatile unsigned short *)(base + 0x1e440) |= 0x20;
33
```

Figure 20. OneNand initialization..\Blackberry\Bootloaders revision 3779 - UFED 1.1.9.0\Sources\Generic\flash.c

66. Cellebrite function `flash_init` is compiled into the Cellebrite BlackBerry extraction payload. The code at line 29 corresponds to the disassembly of the binary content of a Cellebrite extraction payload shown in Figure 21. Just like the code at line 29 of `flash.c`, the assembly code in Figure 21applies an "or" operation to the value `0x20` and the value `0x1E440`.

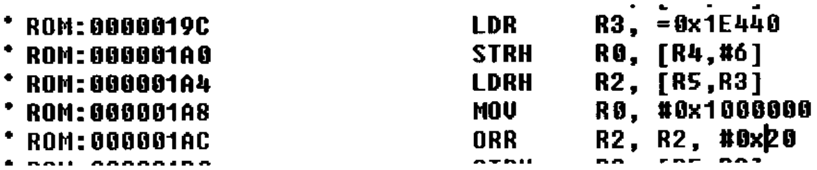


```
• ROM:00000A40          STMFD   SP!, {R4-R6,LR}
• ROM:00000A44          LDR     R4, =loc_4B8
• ROM:00000A48          ADD     R1, R0, #0x1E400
• ROM:00000A4C          ADD     R4, R4, R9
• ROM:00000A50          STR     R0, [R4,#8]
• ROM:00000A54          LDRH    R2, [R1,#0x40]
• ROM:00000A58          ORR     R2, R2, #0x20
```

Figure 21. OneNand command in Cellebrite extraction payload

67. According to Cellebrite developers, the operation at line 29 is from development on previous devices, and has no purpose in the BlackBerry extraction. Even though the

operation at line 29 is unnecessary, the same operation is performed in the MSAB BlackBerry extraction payload, as shown in the disassembly of `xry_loader` in Figure 22. Similar to the Cellebrite code, the MSAB extraction payload performs an "or" operation with the value `0x20` and the value `0x1E440`.



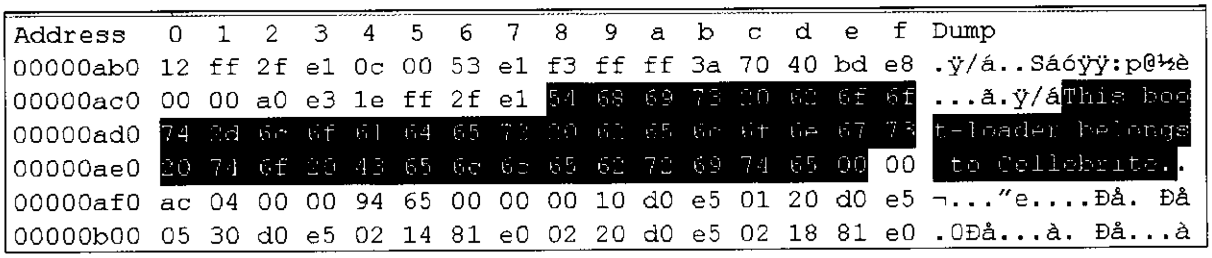
```
• ROM:0000019C                 LDR     R3, =0x1E440
• ROM:000001A0                 STRH    R0, [R4,#6]
• ROM:000001A4                 LDRH    R2, [R5,R3]
• ROM:000001A8                 MOV     R0, #0x1000000
• ROM:000001AC                 ORR     R2, R2, #0x20
```

Figure 22. OneNand command in ..\MSAB XRY 6.5.0\Forensic Pack\BootLoaders\Blackberry\xry_loader

68. It is unusual for software to include unnecessary operations and extremely unusual for two independent development efforts to both include the same unnecessary operation. It is also unlikely that developers would knowingly copy unnecessary operations. Thus, it is very likely that the MSAB developers directly copied the actual expression of Cellebrite's BlackBerry extraction payload, without even understanding the details of what they were copying.

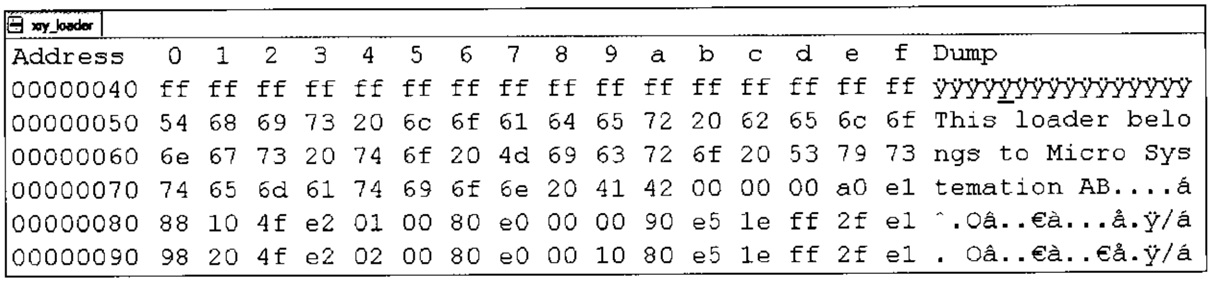
7. OWNERSHIP STATEMENT

69. The Cellebrite extraction payloads include a string stating that they are proprietary Cellebrite files, "`This boot-loader belongs to Cellebrite`" as shown in Figure 23. This is not a common way to state ownership of a program. I typically see the word Copyright followed by a company's name and then a date. Therefore, it is strange that the MSAB extraction payload, which I extracted from the MSAB's XRY Forensic pack 6.5.0, contains a very similar statement, "`This loader belongs to Micro Systemation AB`" as shown in Figure 19. Particularly in the context of the many other indications of copying, these oddly similar ownership statements are another indication that MSAB copied from Cellebrite.



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000ab0  12 ff 2f e1 0c 00 53 e1 f3 ff ff 3a 70 40 bd e8 .ÿ/á..Sáóÿÿ:p@½è
00000ac0  00 00 a0 e3 1e ff 2f e1 54 68 69 73 20 62 6f 6f ...ã.ÿ/áThis boo
00000ad0  74 2d 6c 6f 61 64 65 72 20 62 65 6c 6f 6e 67 73 t-loader belongs
00000ae0  20 74 6f 20 43 65 6c 6c 65 62 72 69 74 65 00 00  to Cellebrite..
00000af0  ac 04 00 00 94 65 00 00 00 10 d0 e5 01 20 d0 e5 ¬..."e....Ðå. Ðå
00000b00  05 30 d0 e5 02 14 81 e0 02 20 d0 e5 02 18 81 e0 .0Ðå...à. Ðå...à
```

Figure 23. Cellebrite ownership statement



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000040  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000050  54 68 69 73 20 6c 6f 61 64 65 72 20 62 65 6c 6f This loader belo
00000060  6e 67 73 20 74 6f 20 4d 69 63 72 6f 20 53 79 73 ngs to Micro Sys
00000070  74 65 6d 61 74 69 6f 6e 20 41 42 00 00 00 a0 e1 temation AB....á
00000080  88 10 4f e2 01 00 80 e0 00 00 90 e5 1e ff 2f e1 ˆ.Oâ..€à...å.ÿ/á
00000090  98 20 4f e2 02 00 80 e0 00 10 80 e5 1e ff 2f e1 . Oâ..€à..€å.ÿ/á
```

Figure 24. MSAB ownership statement

V. CONCLUSION

70. Cellebrite spent considerable time and effort developing their Samsung and BlackBerry solutions for extracting data from the respective smartphones. Cellebrite developed their extraction techniques and software based upon countless hours of painstaking research. During my analysis of MSAB's files, I found numerous similarities with Cellebrite's code and even Cellebrite's name. Individually, the existence of Cellebrite's name, the addresses and commands that Cellebrite selected, Cellebrite's signatures, Cellebrite's parameters, Cellebrite's stack changer function, or even the oddly similar ownership statement within MSAB's code, would strongly indicate that MSAB copied from Cellebrite. However, when evaluated together, it is clear that MSAB directly copied the techniques and expressions of Cellebrite's Samsung and BlackBerry solutions. This collection of similarities is one of the most extensive that I have observed between binary files in my career as an expert witness in software intellectual property litigation. When made available, an analysis of the MSAB XRY source code will undoubtedly confirm that MSAB copied from Cellebrite.

71. It is my understanding that this case is at its initial stages and that discovery in this case will be undertaken shortly. Accordingly, I reserve the right to supplement or amend my opinions in light of any additional evidence, testimony, or information that may be provided to me after the date of this report. I also reserve the right to supplement or amend my opinions in response to any expert reports served by any other party in the lawsuit.

Dated: August 15, 2013

Robert Zeidman

Exhibit A: Resume of Robert Zeidman

Bob Zeidman

ZEIDMAN CONSULTING

15565 Swiss Creek Lane
Cupertino, CA 95014
Tel (408) 741-5809
Fax (408) 741-5231
Email Bob@ZeidmanConsulting.com
Website www.ZeidmanConsulting.com

PROFESSIONAL SUMMARY

Bob Zeidman has management experience in the founding and daily operation of various high-tech companies as well as hands-on experience designing, analyzing, and reverse-engineering hardware and software. Mr. Zeidman is considered a pioneer in the fields of analyzing and synthesizing software source code. Mr. Zeidman is also considered one of the leading experts in the Verilog hardware description language as well as ASIC and FPGA design. He has written several engineering texts and regularly teaches courses in these areas at conferences throughout the world. Mr. Zeidman is also an experienced and well-regarded expert in intellectual property disputes. Bob is certified in the use of CodeSuite®. He holds a B.A. in Physics and a B.S. in Electrical Engineering from Cornell University and a master's degree in Electrical Engineering from Stanford University.

EXPERIENCE

10/1987 - present: Zeidman Consulting

- Company founder and president.
- Provide engineering hardware and software design services to companies.
- Provide engineering support and expert witnesses for high-tech litigation.

8/2007 - present: Software Analysis and Forensic Engineering Corporation

- Company founder and president.
- Software tools for intellectual property litigation.
- Created patented CodeSuite® software including BitMatch®, CodeCLOC®, CodeCross®, CodeDiff®, CodeMatch®, and SourceDetective® for efficiently finding correlation between source code files of different programs.
- Architected CodeGrid for running CodeSuite® on a supercomputer grid.

1/2002 - present: Zeidman Technologies

- Company founder and president.
- Software tools for embedded software development.
- Created patented SynthOS™ software that automatically synthesizes source code for a real time operating system.
- Created patented Molasses® virtualization software that enables a slow speed hardware emulator or prototype to be attached to a high-speed network to emulate network hardware in a live system.

1/2003 – 4/2008: Advisor to Semizone.com

- Web-based training for engineers.
- Advisor on issues relating to e-learning content, development, presentation, and delivery.
- Instructor for various electrical engineering courses.

1/1999 - 12/2002: The Chalkboard Network

- Company founder and President.
- Web-based training for engineers and business professionals.

- Developed Depth Control®, a unique instructional design methodology for Web-based training.

1/1997 - 12/1997: Consultant to Apple Computer

7/1995 - 9/1995

- Firmware development for a multimedia projection system.
- Firmware development for the Apple Studio Display, an advanced flat panel monitor.

7/1996 - 12/1996: Consultant to Hitachi Computer Products (America)

- Helped define an architecture and functional specification for an ATM switch.

10/1995 - 6/1996: Consultant to Cisco Systems

3/1995 - 10/1995

10/1994 - 1/1995

- Wrote Verilog behavioral models and developed a Verilog simulation environment for a 10baseT switch.
- Performed schematic capture, Verilog design, and simulation of an FDDI hub.
- Performed schematic capture, Verilog design, and simulation of an ATM router.

1/1992 - 4/1999: eVAULT Remote Backup Service

- Founded and ran the company.
- Invented the concept of remote backup.
- Set up a remote backup system with a central file server and communication lines including ISDN.
- Wrote remote backup software for DOS, Windows, and OS/2 including a GUI, a backup scheduler and file compression, encryption, and communication routines.

5/1997 - 6/1997: Consultant to Quickturn Design Systems

3/1995 - 7/1995

1/1994 - 2/1994

5/1993 - 6/1993

- Designed custom memory boards for emulation of a supercomputer.

7/1993 - 3/1994: Consultant to Adaptive Video

- Managed a team designing DSP-based medical imaging boards.

1/1993 - 3/1993: Consultant to Wireless Access

- Supervised the design of an FPGA-based prototype for a telecommunications encoding and decoding scheme.

6/1991 - 9/1991: Consultant to STEP Engineering

- Reviewed and optimized RISC hardware and software.

12/1990 - 8/1991: Consultant to RICOH Corporation

- Led a team that designed four RISC-based controllers for laser printers, scanners, faxes, and copiers.

9/1989 - 11/1990: Consultant to DAVID Systems

- Designed network interface boards for 10BaseT, 10Base2, and FOIRL.
- Led an international team that tested an AMI encoding scheme and designed an ASIC for a digital phone set.

5/1993 - 8/1993: Consultant to IKOS Systems

8/1992 - 2/1993

3/1988 - 8/1989

- Designed a high speed SBUS to MXIbus interface and a high-speed controller for a parallel processor.
- Architected, designed, and wrote diagnostic software for a RISC-based controller for a simulation accelerator.

10/1987 - 10/1988: Consultant to Stanford University

- Led a team of graduate students under Professor Mike Flynn in the design and testing of a neural network memory.

1/1986 - 3/1988: Telestream Corporation

- Developed an architecture model for a telecom parallel processor system.
- Modeled, simulated, and tested a proprietary bus.
- Designed a telecom processing element ASIC.
- Designed a telecom system backplane.

9/1985 - 12/1985: American Supercomputers

- Designed a data cache and the register section logic for a CRAY compatible supercomputer.
- Assisted with the implementation of a behavioral simulator.

1/1985 - 8/1985: ROLM Corporation

- Simulated, debugged, and tested cache and memory control ASICs for a minicomputer.
- Microcoded the character string instructions.

4/1983 - 12/1984: Signetics Corporation

- Project leader for a CMOS DMA Interface chip (68431).
- Simulated a TTL VME bus controller chip (68172).
- Redesigned and simulated an oxide isolated ISL Enhanced Video Attribute Controller chip (2675T).
- Simulated a TTL Disk Phase Locked Loop chip (68459).
- Created a standard procedure for developing ASICs.

LEGAL CONSULTING

4/2013 – present: Audionlcs System v. AAMP of Florida

Law Firm: Trojan Law Offices
Client: Audionlcs System
Court: U.S. District Court, Central District of California, Western Division
Case: 2:12-cv-10763-MMM-JEM

- Alleged patent infringement of aftermarket stereo interface devices.
- Wrote an expert report.
- Testified in deposition.

10/2012 – 12/2012: E-Tech USA v. 2lemetry

Law Firm: Reed Smith
Client: 2lemetry
Court: U.S. District Court, District of Colorado
Case: cv-03371-REB-KMT

- Alleged trade secret misappropriation of cloud platform software.
- Compared software source code using CodeMatch.

- Wrote an expert report.

7/2012 – present: Kenneth C. Henry v. Petrolink v. Digital Well File

Law Firms: Wright & Close; Martin, Disiere, Jefferson & Wisdom
Client: neutral expert
Court: District Court of Harris County, Texas, 133rd Judicial District
Case: 2010-08178

- Alleged copyright infringement of real-time decision making software for oilfield data.
- Alleged trade secret theft.
- Compared software source code using CodeMatch.
- Wrote two expert reports.

7/2012 – present: Elan Microelectronics v. Pixcir Microelectronics

Law Firm: Alston & Bird LLP
Client: Elan Microelectronics
Court: U.S. District Court, District of Nevada
Case: 2:10-cv-00014-GMN-(PAL)

- Alleged patent infringement of capacitive touch controller hardware and software.

1/2012 – present: ThinkOptics v. Nintendo of America, et al.

Law Firm: Nix Patterson & Roach LLP
Client: ThinkOptics
Court: U.S. District Court, Eastern District Of Texas, Tyler Division
Case: 6:2011cv00454

- Alleged patent infringement of video game controller hardware.

1/2012 – 9/2012: E-Micro Corporation v. Google, et al.

Law Firm: Nix Patterson & Roach LLP
Client: E-Micro
Court: U.S. District Court, Eastern District Of Texas, Tyler Division
Case: 6:11-CV-465 (JDL)

- Alleged patent infringement of mobile commerce software.

7/2011 – 12/2011: Facebook v. Power Ventures

Law Firm: Orrick, Herrington & Sutcliffe
Client: Facebook
Court: U.S. District Court, Northern District of California
Case: 5:08-cv-05780 JW (HRL)

- Alleged copyright infringement of social network software.
- Alleged trademark infringement of social network software.
- Alleged DMCA violation.
- Alleged violation of the CAN-SPAM Act.
- Alleged violation of the the Computer Fraud and Abuse Act ("CFAA").
- Alleged violation of the California Comprehensive Computer Data Access and Fraud Act.
- Alleged unfair competition.
- Wrote expert report.

4/2011 – present: Motorola Mobility, Google v. Microsoft

Law Firm: Sidley Austin
Client: Microsoft
Court: U.S. District Court, Southern District of Florida
Case: 1:10-cv-2406J-MORENO

- Alleged patent infringement of voicemail software.
- Alleged patent infringement of graphics software.

- Analyzed patents.
- Examined source code.
- Wrote expert reports.
- Testified in deposition.

8/2010 – 3/2011: Cross Match Technologies v. Suprema and Mentalix

Law Firm: Latham & Watkins
Client: Cross Match Technologies
Court: U.S. International Trade Commission
Investigation: 337-TA-720

- Alleged patent infringement of fingerprint scanning software.
- Analyzed patents.
- Examined source code.
- Compared software source code using CodeMatch.
- Wrote an expert report.
- Testified in deposition.

7/2010 – 4/2011: Xpoint Technologies v. Symantec

Law Firm: Quinn Emanuel Urquhart & Sullivan
Client: Symantec
Court: U.S. District Court, District of Delaware
Investigation: 09-CV-0026 (SLR)

- Alleged patent infringement of backup software.
- Analyzed patents.
- Examined software.
- Wrote an expert report.

4/2010 – 7/2012: Brocade v. A10 Networks

Law Firms: Orrick, Herrington & Sutcliffe; McDermott, Will & Emery
Client: Brocade
Court: U.S. District Court, Northern District Of California, San Jose Division
Case: C 10-03428 LHK (PSG)

- Alleged copying of software for network controllers.
- Compared software source code using CodeCross, CodeDiff, and CodeMatch.
- Measured software development using CodeCLOC.
- Wrote an expert report and several declarations.
- Testified in deposition.
- Testified at trial.

3/2010 – 9/2010: Datamaxx v. Computer Products of Illinois

Law Firm: Ausley & McMullen
Client: Computer Products of Illinois
Court: U.S. District Court, Northern District Of Florida, Tallahassee Division
Case: 4:09cv435-RH/WCS

- Alleged copying of software to enable law enforcement.
- Compared software source code using CodeMatch.
- Wrote a declaration and a rebuttal expert report.
- Testified in deposition.

2/2010 – 7/2010: SplitFish v. Bannco

Law Firm: Finnegan, Henderson, Farabow, Garrett & Dunner, LLP
Client: SplitFish
Court: U.S. District Court, Northern Easter District of Virginia, Alexandria Division
Case: l:10cv297

- Preliminary injunction against defendant granted.
- Alleged copying of firmware for video game controllers.
- Compared software source code using BitMatch.
- Wrote two declarations.

12/2009 – 7/2013: Robin Antonick v. Electronic Arts

Law Firm: Keker & Van Nest LLP
Client: Electronic Arts
Court: U.S. District Court, Northern District of California
Case: 3:11-cv-01543-CRB (EDL)

- Contract dispute.
- Compared software source code using CodeMatch.
- Wrote expert reports.
- Testified in deposition.
- Testified in court.

0/2009-11/2009: Sigma Six Technologies and Sigma Six Consulting. v. Nagarro and T-Systems Enterprise Services.

Law firm: Ropers, Majeski, Kohn & Bentley PC
Client: Nagarro
Court: U.S. District Court, Northern District of California
Case: C 08-05633 JW

- Case was settled.
- Alleged trade secret infringement case involving enterprise client user interface software.

8/2009 – 11/2010: Zynga v. Green Patch

Law firm: Quinn Emanuel Urquhart & Sullivan
Client: Zynga
Court: U.S. District Court, Northern District of California
Case: CV-09-3636 SC (EMC)

- Case was settled.
- Alleged copyright infringement case involving web-based social networking game software.
- Compared software source code using CodeMatch.
- Wrote two declarations and an expert report.

8/2009 – 5/2012: De Lage Landen Operational Services v. Third Pillar Systems

Law Firm: Wilson Sonsini Goodrich & Rosati
Client: Third Pillar Systems
Court: U.S. District Court, Eastern District of Pennsylvania
Case: 09-cv-02439-HB

- Case was adjudicated.
- Alleged trade secret misappropriation and contract dispute involving lending and leasing software.
- Compared software source code using CodeMatch and CodeDiff.
- Wrote an expert report and a rebuttal expert report.
- Testified in deposition.
- Testified at trial.

2/2009 – 10/2009: Minden Schipper & Associates v. Cancercare Manitoba (Varian Medical Systems)

Client: Varian Medical Systems
Court: Queen's Bench, Winnipeg Centre, Canada
Case: CI 05-01-45377

- Case was settled.

- Alleged trade secret misappropriation and copying of oncology diagnosis expert system software.
- Compared software source code using CodeMatch.
- Wrote an expert report.

11/2008 – 1/2010: Applied Materials v. Advanced Micro-Fabrication Equipment Co.

Law firm: Goodwin Procter
Client: Applied Materials
Court: U.S. District Court, Northern District Of California
Case: C07 05248 JW (PVT)

- Case was settled.
- Alleged copying of software for semiconductor manufacturing machines.
- Compared software source code using CodeMatch.
- Wrote expert reports.
- Testified in deposition.

9/2008 – 9/2009: Abanco Investments v. GuestLogix

Law firm: Patterson, Thuente, Skaar & Christensen
Client: GuestLogix
Court: U.S. District Court, Northern District of Illinois
Case: 07 C 1071

- Case was settled.
- Alleged trade secret theft involving point-of-sale software.
- Compared software source code using CodeMatch.
- Wrote an expert report.

9/2008 – 3/2010: Personnel Department v. CareerBuilder

Law firm: Jones, Day, Reavis & Pogue
Client: CareerBuilder
Court: U.S. District Court, District of Vermont
Case: 2:08-cv-59-wks

- Case was settled.
- Alleged trade secret theft involving resume building software.
- Compared software source code using CodeMatch.
- Wrote an expert report.

9/2008 - 12/2008: Honeywell, Metrologic, OmniPlanar v. Datalogic

Law firm: Robins, Kaplan, Miller & Ciresi
Client: Omniplanar
Court: U.S. District Court, District of New Jersey
Case: 1:2008cv05234

- Case was settled.
- Alleged copyright infringement and trade secret theft involving bar code scanner firmware.
- Compared software source code to software binary code using BitMatch.

7/2008-10/2008: Facebook v. StudiVZ

Law firm: Orrick, Herrington & Sutcliffe
Client: Facebook
Court: U.S. District Court, Northern District of California
Case: 5:08-CV-03468 JF

- Case was settled.
- Alleged copyright infringement case involving web-based social networking software.
- Compared software source code using CodeMatch.

4/2008 – 12/2010: Esbin & Alter v. Zappier, et al.

Law firm: Alter & Alter
Client: Esbin & Alter
Court: U.S. District Court, Southern District of New York
Case: 08 Civ. 313

- Contract dispute case involving billing and document management software.
- Compared software source code using CodeMatch.
- Wrote two declaration and two expert reports.

4/2008 – 12/2008: Piper Jaffray v. Vermilion Capital Management

Law firm: Patterson, Thuente, Skaar & Christensen
Client: Vermilion Capital Management
Court: Minnesota District Court, Fourth Judicial District, County of Hennepin
Case: 07-20203

- Alleged trade secret case involving stock market technical analysis software.
- Compared software source code using CodeCross, CodeDiff, and CodeMatch.
- Wrote an expert report.

4/2008 – 11/2008: Intelligraphics v. Marvell Semiconductor

Law firm: Sommers and Schwartz
Client: Intelligraphics
Court: U.S. District Court, Northern District Of California – San Francisco Division
Case: C-07-2499 JCS

- Contract dispute case involving WLAN firmware and drivers.
- Compared software source code using CodeDiff.
- Wrote an expert report.
- Testified at deposition.

3/2008 – 8/2008: Optovue v. Carl Zeiss Meditec

Law firm: Nixon Peabody
Client: Carl Zeiss Meditec
Court: U.S. District Court, Northern District of California - Oakland Division
Case: C 07-03010 CW

- Case was settled.
- Alleged copyright/trade secret theft case involving optical coherence tomography software.
- Compared software source code using CodeMatch.
- Wrote an expert report.

2/2008 – 4/2008: Gemstar v. Digeo

Law firm: Ropes & Gray
Client: Gemstar
Court: U.S. District Court, Central District of California – Western Division
Case: CV-06-6519

- Case was settled.
- Alleged patent infringement case involving program guide displays.
- Wrote an expert report.

1/2008 – present: MSC Software v. Altair Engineering, et al.

Law firm: Dykema Gossett
Client: MSC Software
Court: U.S. District Court, Eastern District of Michigan – Southern Division
Case: 2:07-cv-12807

- Alleged trade secret theft case involving motion simulation software.

- Compared software source code using CodeMatch.
- Wrote a declaration and an expert report.
- Testified at a hearing.
- Testified in deposition.

5/2007 – 12/2008: The MathWorks v. COMSOL

Law firm: Jones, Day, Reavis & Pogue
Client: The MathWorks, Inc.
Court: U.S. District Court, Eastern District of Texas – Tyler Division
Case: 6:06-CV-335

- Case was settled.
- Alleged copyright infringement case involving mathematical modeling software.
- Compared software source code using CodeMatch.

5/2007 – 12/2008: The MathWorks v. COMSOL

Law firm: Jones, Day, Reavis & Pogue
Client: The MathWorks, Inc.
Court: U.S. District Court, Eastern District of Texas – Tyler Division
Case: 6:06-CV-334

- Case was settled.
- Alleged patent infringement case involving mathematical modeling software.

5/2007 – 6/2007: Third Party Verification v. SignatureLink

Law firm: Law Offices of Brian S. Steinberger
Client: Third Party Verification
Court: U.S. District Court, Middle District of Florida
Case: 6:06-cv-00415

- Case was settled.
- Alleged copyright infringement case involving web-based signature capture software.
- Compared software source code using CodeMatch.
- Wrote an expert report.

4/2007 – 7/2008: Symantec v. Commissioner of Internal Revenue

Law firm: Baker & McKenzie
Client: Symantec
Court: U.S. Tax Court
Case: 12075-06

- Case was adjudicated.
- Software tax dispute
- Software comparison using CodeDiff.
- Wrote an expert report.
- Testified at trial.

4/2007 – 5/2007: Kernius & Frise v. International Electronics

Law firm: Zito tlp
Client: Kernius & Frise
Case: 05-CV-1927
Court: U.S. District Court, District of Maryland

- Alleged patent infringement case involving modems and call waiting.
- Wrote an expert report.
- Testified in deposition.

2/2007 – 8/2008: Quantum Research Group (Atmel) v. Apple, Cypress, Fingerworks

Law firm: Zito tlp, Sidley Austin

Client: Quantum Research Group (Atmel)
Court: U.S. District Court, District of Maryland
Case: 05-cv-03408-WMN

- Alleged patent infringement case involving capacitive sensing devices.
- Assisted with claim construction.
- Wrote an expert report.
- Testified in deposition.

8/2006 – 8/2007: AdTech RFID v. Adept Identification Technologies, et al.

Law firm: Wilson Sonsini Goodrich & Rosati
Client: Adept Identification Technologies
Court: Superior Court of Santa Clara County California
Case: 1:06-CV-057464

- Case was settled.
- Alleged trade secret theft case involving RFID software.
- Compared software source code using CodeMatch.
- Wrote an expert report.

6/2006 - 2/2007: Medinformatix v. AcerMed

Law firm: Timothy McGonigle
Client: MedInformatix
Arbitration: JAMS
Ref: 1220035252

- Case was decided at arbitration.
- Alleged trade secret case involving electronic medical records software.
- Assisted with determination of trade secrets.
- Compared software source code using CodeMatch.
- Wrote declarations.
- Testified in deposition.

5/2006 - 11/2006: Iconix v. netPickle, et al.

Law firm: Orrick, Herrington & Sutcliffe
Client: netPickle
Court: U.S. District Court, Northern District of California – Oakland Division
Case: 4:06-cv-02201

- Case was settled.
- Alleged copyright infringement case involving web-based presentation software.
- Compared software source code using CodeDiff.
- Wrote a declaration.
- Wrote an expert report.
- Testified in deposition.

4/2006 - 10/2006: Forgent v. Microsoft et al.

Law firm: Susman Godfrey
Client: Forgent
Court: U.S. District Court, Northern District of California – San Jose Division
Case: M:05-CV-01654

- Case was settled.
- Alleged patent infringement case involving JPEG encoding of pictures in files.
- Reverse engineered video equipment.

2/2006 – 5/2008: Rasterex Holdings v. Research in Motion

Law firm: Kilpatrick Stockton
Client: Rasterex Holdings

Court: Superior Court of Fulton County, State of Georgia
Case: 2003-cv-76785

- Case was settled.
- Alleged copyright infringement case involving mobile document translation and storage software.
- Compared software source code using CodeMatch.
- Wrote an expert report.
- Testified in deposition.

2/2006 - 9/2006: Medinformatix v. Camtronics Medical Systems

Law firm: Timothy McGonigle
Client: MedInformatix
Court: U.S. District Court, Central District of California
Case: 2:05-cv-04829 SJO

- Case was settled.
- Alleged trade secret theft case involving electronic medical records software.
- Assisted with determination of trade secrets.
- Wrote declarations.

8/2005 – 12/2007: Moneygram Payment Systems v. Enterprise Payment Solutions

Law firm: Michael Best & Friedrich LLP
Client: Moneygram
Court: U.S. District Court, Eastern District of Tennessee
Case: 1:05-cv-00172

- Alleged copyright infringement case involving ACH financial software.
- Compared software source code using CodeMatch.
- Wrote an expert report.

8/2005 - 8/2006: Silvaco v. Specular

Law firm: Wilson Sonsini Goodrich & Rosati
Client: Specular
Court: Superior Court of Santa Clara County California
Case: 1:04-cv-031951

- Case was settled.
- Alleged trade secret case involving electronic design automation (EDA) software.
- Compared software source code using CodeMatch.
- Wrote an expert report.

7/2005-2/2008: ConnectU v. Facebook, et al.

Law firm: Orrick, Herrington & Sutcliffe
Client: Facebook
Court: U.S. District Court, District of Massachusetts
Case: 1:04-cv-11923

- Case was settled.
- Alleged copyright infringement case involving social network software.
- Compared software source code using CodeMatch.

4/2005 – 3/2007: Merchant Transaction Systems, et al. v. Nelcela, et al.

Law firm: Lewis & Roca
Client: Merchant Transaction Systems/POST Integration/Ebocom
Court: U.S. District Court, District of Arizona
Case: 2:02-cv-01954

- Alleged copyright infringement involving credit card processing software.
- Compared software source code using CodeMatch.
- Wrote an expert report.

- Testified in deposition.

4/2005: Brod v. Lev, et al.
Law firm: Wilson Sonsini Goodrich & Rosati
Client: Lev
Court: Superior Court of Santa Clara County California
Case: 1:03-cv-005813

- Case was dismissed.
- Alleged copyright infringement case involving Internet acceleration software.
- Compared software source code using CodeMatch.

3/2005 - 12/2005: American Video Graphics v. Electronic Arts, et al.
Law firm: McKool Smith
Client: AVG
Court: U.S. District Court, Eastern District of Texas – Tyler Division
Case: 6:04-CV-398

- Case was settled.
- Alleged patent infringement case involving 3D graphics software algorithms.
- Examined source code for over 50 video games.
- Wrote claim charts for each video game.

3/2005 - 12/2005: American Video Graphics v. Sony Corporation of America, et al.
Law firm: McKool Smith
Client: AVG
Court: U.S. District Court, Eastern District of Texas – Tyler Division
Case: 6:04cv399

- Case was settled.
- Alleged patent infringement case involving 3D graphics hardware algorithms.
- Examined graphic chips.

8/2004 – 12/2007: Creative Science Systems v. Forex Capital Markets
Law firm: Baker & McKenzie/Sommers and Schwartz
Client: Creative Science Systems
Court: U.S. District Court, Northern District of California
Case: 5:04-cv-03746

- Case was settled.
- Alleged copyright infringement involving web-based financial software.
- Compared software source code using CodeMatch.
- Compared software object code.
- Wrote a declaration and an expert report
- Testified in deposition.

8/2004 - 3/2005: XIOtech v. Compellent Technologies, et al.
Law firm: Faegre & Benson
Client: Compellent
Court: Minnesota District Court, Fourth Judicial District, County of Hennepin
Case: 04-5065

- Case was settled.
- Alleged trade secret theft involving storage area network (SAN) software.
- Compared software source code using CodeMatch.
- Compared features and researched prior art for storage area network (SAN) software.
- Wrote an expert report.

8/2004 – 9/2004: OpenTable v. Smart Restaurant Solutions

Law firm: Wilson Sonsini Goodrich & Rosati
Client: Smart Restaurant Solutions
Court: Superior Court of the State of California for the County of San Francisco
Case: CGC-03-424516

- Judgment at trial.
- Alleged copyright infringement involving restaurant management software.
- Compared software source code using CodeMatch.
- Wrote an expert report

7/2004 - 2/2005: Zoran and Oak Technology v. MediaTek, et al.

Law firm: Wilson Sonsini Goodrich & Rosati /Hogan & Hartson
Client: MediaTek
Court: U.S. International Trade Commission
Investigation: 337-TA-506

- Case was settled.
- Alleged patent infringement involving CD-ROM/DVD controller hardware.
- Examined VHDL for two CD-ROM/DVD controller chips to determine their architectures and implementations.
- Wrote an expert report and created exhibits for trial.
- Testified in deposition.
- Testified at trial.

4/2004 - 11/2004: Agere Systems v. Intersil

Law firm: Kirkland & Ellis
Client: Agere
Court: U.S. District Court, Eastern District of Pennsylvania
Case: 02-CV-08219, 02-CV-1544

- Case was settled.
- Alleged copyright infringement and contract dispute involving WLAN chips.
- Compared firmware source code using CodeMatch.
- Examined Verilog and VHDL source code.
- Wrote an expert report and rebuttal to opposition expert report.

4/2003 - 5/2003: Alvis v. Hewlett-Packard

Law firm: Drinker Biddle & Reath
Client: Hewlett Packard
Court: U.S. District Court, District Court of Jefferson County, Texas
Case: A-164,880

- Case was settled.
- Class action suit involving reliability of floppy disk drives and software patches.
- Wrote an expert report.
- Testified in deposition.

4/2003 - 5/2003: MediaTek Software Clean Room Development Project

Law firm: MacPherson Kwok
Client: MediaTek

- Clean room code development.
- Reviewed source code and compared different source code routines for similarities.

3/2003 - 5/2003: Research In Motion v. Good Technology

Law firm: Jones, Day, Reavis & Pogue
Client: Research In Motion

Court: U.S. District Court, District of Delaware
Case: 02-556-JJF, 02-1286-JJF, 02-1338-JJF

- Case was settled.
- Alleged patent infringement case involving handheld wireless devices and supporting software.
- Analyzed software source code.
- Assisted with deposition of opposing expert.
- Wrote claim charts.

8/2001 - 9/2001: Intel v. VIA Technologies

Law firm: Howrey Simon Arnold & White
Client: Intel
Court: U.S. District Court, Northern District of California
Case: C99-03062

- Case was settled.
- Alleged patent infringement case involving computer motherboards.
- Examined computer motherboards for patent infringement.

7/2001 - 4/2003: Intel v. VIA Technologies

Law firm: Dewey Ballantine/Brobeck, Phleger & Harrison
Client: Intel
Court: U.S. District Court, Western District of Texas
Case: A-01-CA-602-SS

- Case was settled.
- Alleged patent infringement cases involving CPUs and computer chipsets.
- Assisted with claim construction.
- Wrote test case software in assembly language.
- Examined computer motherboards.
- Analyzed Verilog code of CPUs.
- Wrote several expert reports.
- Wrote several claim charts.

3/2001 - 3/2001: KRS Distributing v. Gatten Insurance

Law firm: Stone & Hiles
Client: Gatten Insurance

- Insurance claim.
- Examined a fax machine to retrieve stored documents.

7/1997 - 3/1999: Texas Instruments v. Hyundai Electronics Industries Co.

Law firm: Jones, Day, Reavis & Pogue
Client: Texas Instruments
Court: U.S. District Court, Eastern District of Texas
Case: 2:98CV74

- Case was settled.
- Alleged patent infringement involving semiconductor wafer handling hardware, software, and communication protocols.
- Reverse engineered hardware and software in order to determine infringement.
- Constructed exhibits.
- Assisted with the writing of expert reports.
- Assisted with the writing of claim charts.

8/1996 - 12/1996: Texas Instruments v. Samsung Electronics, et al.,

Law firm: Jones, Day, Reavis & Pogue
Client: Texas Instruments
Court: U.S. District Court, Eastern District of Texas

Case: 2:96-CV-1, 2:96-CV-2

- Case was settled.
- Alleged patent infringement involving semiconductor wafer handling hardware, software, and communication protocols.
- Reverse engineered hardware and software in order to determine infringement.
- Constructed exhibits.
- Assisted with the writing of expert reports.
- Assisted with the writing of claim charts.

2/1996 - 7/1996: Cirrus Logic v. Agarwal, et al.

Law firm: Morrison & Foerster
Client: Cirrus Logic
Court: Superior Court of Santa Clara County California
Case: CV 745373

- Case was settled.
- Alleged trade secret case involving semiconductors and LCD display technology.
- Examined validity of trade secrets.
- Reconstructed the history of an internal engineering project.
- Researched prior art.

HONORS, AWARDS, AND DISTINCTIONS

Engineering and Science

- Final Round, 2011 Jolt Awards, for the book *The Software IP Detective's Handbook: Measurement, Comparison, and Infringement Detection.*
- Outstanding Engineer in a Specialized Field: For Innovative Contributions in the Area of Forensic Software Analysis, IEEE Santa Clara Valley Section, 2010.
- Session's Best Paper Award, The11th World Multi-Conference on Systemics, Cybernetics and Informatics, 2007.
- The Number 5 Programmable Logic "How To" article of 2006, Programmable Logic DesignLine newsletter.
- Finalist, Design News magazine 2006 Golden MouseTrap Award: Design and Development Software Tools, for SynthOS.
- Winner, Software Development magazine 2003 Jolt Reader's Choice Award for the book *Designing with FPGAs and CPLDs.*
- Senior Member, IEEE
- Top PLD/FPGA News and Feature Article for 2003, CMP Media
- Winner, Wyle/EETimes American By Design Contest, 1994
- Stanford Graduate Engineering Fellowship
- Eta Kappa Nu (Electrical Engineering honor society)
- Association for Educational Data Systems Honorable Mention
- Bausch & Lomb Honorary Science Award

Writing and Filmmaking

- Indie Excellence 2013 Winner, Humor category, for the novel *Good Intentions.*
- Indie Excellence 2013 Finalist, Political Thriller category, for the novel *Good Intentions.*
- Honorable Mention, 2013 San Francisco Book Festival, for the novel *Horror Flick.*
- Pinnacle Book Achievement Award 2012 for the novel *Good Intentions.*
- Semifinalist, November 2011 Amazon Studios Best Kids and Family Script Award, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Semifinalist, October 2011 Amazon Studios Best Script Award, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."

- Semifinalist, September 2011 Amazon Studios Best Script Award, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Semifinalist, August 2011 Amazon Studios Best Script Award, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Semifinalist, July 2011 Amazon Studios Best Script Award, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Semifinalist, June 2011 Amazon Studios Best Sci-Fi/Action Script Award, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Semifinalist, June 2011 Amazon Studios Best Script Award, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Semifinalist, 2004 Cinequest Screenwriting Competition, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Second Place, 2002 Autumn Moon Productions Screenplay Awards, for the screenplay "Horror Flick."
- Third Place, 2002 Autumn Moon Productions Screenplay Awards, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Honorary Mention, 2002 Autumn Moon Productions Screenplay Awards, for the screenplay "Sex and Violence."
- Certificate of Merit, 2002 International Screenplay Competition, for the screenplay "Horror Flick."
- Certificate of Merit, 2002 International Screenplay Competition, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Certificate of Merit, 2002 International Screenplay Competition, for the screenplay "Sex and Violence."
- First Place, 2001 Focus on Writers Contest, for the screenplay "Horror Flick."
- Special Mention Winner, 2001 Screenwriting Showcase Awards, for the screenplay "Horror Flick."
- Finalist, 2001 Empyrion Screenplay Competition, for the screenplay "Sex and Violence."
- Finalist, 2001 New Century Writer Awards, for the screenplay "Horror Flick."
- Top Ten Finalist, 2001 Tennessee Screenwriting Association Competition, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Semifinalist, 2001 WordsFromHere Contest, for the screenplay "Horror Flick."
- Semifinalist, 2001 Venice Arts Screenwriting Competition, for the screenplay "Horror Flick."
- Semifinalist, 2001 National Screenwriting Competition, for the screenplay "Horror Flick."
- Semifinalist, 2001 National Screenwriting Competition, for the screenplay "Sex and Violence."
- Quarterfinalist, 2001 Fade In: Screenwriting Awards, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Quarterfinalist, Texas Film Institute 2001 Screenplay Competition, for the screenplay "Horror Flick."
- Top Finalist, BDR 2000 Productions New Millennium Screenplay 2001 Contest, for the screenplay "Horror Flick."
- Certificate of Merit, Writer's Digest 2000 National Self-Published Book Awards, for the novel "Horror Flick."
- Semifinalist, 2000 poetry.com North American Open Poetry Contest, for the poem "I Remember."
- Quarterfinalist, 1999 New Century Writer Awards, for the novel "Horror Flick."
- Quarterfinalist, 1999 New Century Writer Awards, for the screenplay "The Amazing Adventure of Edward and Dr. Sprechtmachen."
- Third Place, 1998 Magnum Opus Discovery Awards of the C.C.S. Entertainment Group and the Hollywood Film Festival, for the novel "Horror Flick."
- Quarterfinalist, 1998 Empire Screenplay Contest, for the screenplay "Sex and Violence."
- A reading of my screenplay "Sex and Violence" was performed by the Independent Media Artists Group (IMAGE), on August 15th, 1998.
- Vermont Studio Center scholarship to attend a one-month writing retreat, April 1998.
- Semifinalist, 1997 Monterey County Film Contest, for the screenplay "Sex and Violence."

- First Place, 1993 Foster City Annual Writer's Contest, for unpublished short story "The Contest."
- Semifinalist, 1993 national Syndicated Fiction Project, for unpublished short story "The Contest."
- First Place, 1990 Foster City Annual Writer's Contest, for unpublished short story "The Lost and Found Virginity."
- 1989 Philadelphia International Film Festival showing of the short film "Writer's Block."
- First Place, 1988 Fremont Film Festival, for the short film "February 20, 1988."

Miscellaneous

- United Synagogue Award for Excellence
- Biography in Who's Who in America
- University Unions Distinguished Service Award
- Phi Beta Kappa
- Ivy League Honor Society
- Alpha Lambda Delta honor society
- Dual Degree Program, Cornell University
- National Merit Scholarship
- Literary Society Foundation Award Gold Medal, Excellence in German
- Literary Society Foundation Award Bronze Medal, Excellence in German
- Rumsey Scholarship, Cornell Club of Philadelphia
- City of Philadelphia Scholarship
- School District of Philadelphia Scholarship
- Fourth Prize, Colonial Philadelphia Historical Society Essay Contest
- Founder, Delaware Valley Teen Mensa

SPECIAL KNOWLEDGE AND SKILLS

- CodeSuite certified.
- Software source code analysis and synthesis
- Computer architectures: AMD 29000, CRAY XMP, Data General MV8000, IBM PC, IDT R4650, Intel 8051, Intel x86 family, Motorola 68000, 68HC11, 68HC08, 68HC05, TI TMS320Cxx, TMS340xx
- Networking protocols: ATM, Ethernet
- Buses: ADB, EISA I^{2}C, ISA, MXI, PCI, SBUS, VME
- Hardware programming languages: ABEL, AHDL, CUPL, PALASM, Verilog, VHDL
- Software programming languages: APL, BASIC, C, C++, Delphi, FOCAL, FORTRAN, Java, LISP, Pascal, Perl, PHP, PL/1, PowerBuilder, SQL, Visual BASIC, various assembly, machine languages
- Operating systems: AOS/VS, MSDOS, UNIX, VMS, Windows 3.1/NT/9x/2000/XP/Vista
- Workstations: Apple Macintosh, Daisy, IBM PC, Hewlett Packard, SUN, VALID
- Electronic Design Automation (EDA)
- CAD tools: Concept, Futurenet, MAX+Plus II (Altera), MDE (LSI Logic), Mentor, Orcad, PROCapture, P-CAD, Schema, Viewlogic, XACT (Xilinx)
- Simulation accelerators: Mentor Graphics, Cadence Design Systems
- Hardware emulators: Mentor Graphics, Cadence Design Systems
- ASIC design
- FPGA and CPLD design: Actel, Altera, Lattice, Xilinx
- Miscellaneous design experience: Cache memory, telecommunications, data communications, digital signal processing (DSP), digital logic (CMOS, ECL, TTL), analog
- Patent infringement
- Trade secret theft
- Copyright infringement
- Plagiarism detection
- Solid state theory

- Information theory

EDUCATION

Master of Science in Electrical Engineering, 1982, Stanford University

Bachelor of Science with distinction in Electrical Engineering, 1981, Cornell University

Bachelor of Arts cum laude in Physics and with distinction in all subjects, 1981, Cornell University

BOOKS

1. Bob Zeidman, *Just Enough Electronics to Impress Your Friends and Colleagues*, Swiss Creek Publications, Cupertino, CA, 2013, 214pp.
2. Bob Zeidman, *The Software IP Detective's Handbook: Measurement, Comparison, and Infringement Detection*, Prentice-Hall, Upper Saddle River, NJ, 2011, 450pp.
3. Clive Maxfield, *FPGAs: World Class Designs*, Elsevier Inc., Burlington, MA, 2009, Chapter 1 (reprint).
4. Ashby, Baker, Ball, Crowe, Hayes-Gill, Hickman, Kester, Mancini, Grout, Pease, Tooley, Williams, Wilson, Zeidman, C*ircuit Design: Know It All*, Elsevier Inc., Burlington, MA, 2008, Chapters 27-29 (reprint).
5. R. C. Cofer, Clive Maxfield, Bob Zeidman, Richard Munden, Rick Gentile, *Newnes FPGAs: ebook Collection*, Elsevier Science & Technology Books, Burlington, MA, 2008 (reprint).
6. Bob Zeidman, *Designing with FPGAs and CPLDs*, CMP Books, Lawrence, KS, 2002, 220pp.
7. Bob Zeidman, *Introduction to Verilog*, IEEE Press, Piscataway, NJ, 2000, 99pp.
8. Bob Zeidman, *Verilog Designer's Library*, Prentice-Hall, Upper Saddle River, NJ, 1999, 411pp.
9. Bob Zeidman, *Good Intentions*, Swiss Creek Publications, Cupertino, CA, 2012, 259pp.
10. Bob Zeidman, *Horror Flick*, Swiss Creek Publications, Cupertino, CA, 1999, 341pp.
11. Bob Zeidman, *The Amazing Adventure of Edward and Dr. Sprechtmachen*, Swiss Creek Publications, Cupertino, CA, 1998, 73pp.

PAPERS AND PRESENTATIONS

1. Zeidman, Bob, "Did Bill Gates Steal the Heart of DOS?" IEEE Spectrum (http://spectrum.ieee.org/computing/software/did-bill-gates-steal-the-heart-of-dos), July 2012.
2. Zeidman, Bob, "This one really takes the cake—and the schematics," EDN magazine, July 2012.
3. Zeidman, Bob and Kovanis, Evan, "Round 2: Did Oracle Overlook the Smoking Gun in its Case against Google?" *IPWatchdog* (http://www.ipwatchdog.com/2012/07/11/round-2-did-oracle-overlook-the-smoking-gun-in-its-case-against-google/id=26223), July 11, 2012.
4. Kovanis, Evan and Zeidman, Bob, "Did Oracle Overlook the Smoking Gun in its Case against Google?" *IPWatchdog* (http://www.ipwatchdog.com/2012/06/26/did-oracle-overlook-the-smoking-gun-in-its-case-against-google/id=25843), June 26, 2012.
5. Melling, L. and Zeidman, B., "Comparing Android Applications to Find Copying," *Journal of Digital Forensics, Security and Law*, Vol. 7, No. 1, 2012.
6. Zeidman, Bob, "Program Identifiability: How easily can you spot your code?" *Embedded.com* (http://www.embedded.com/design/embedded/4374526/Program-Identifiability--How-easily-can-you-spot-your-code-), June 7, 2012.

7. Zeidman, Bob, "Setting the Record Straight: Patent Trolls vs. Progress" *IPWatchdog* (http://www.ipwatchdog.com/2012/05/01/setting-the-record-straight-patent-trolls-vs-progress/id=24491), May 1, 2012.
8. Baer, N. and Zeidman, B., "Measuring Whitespace Pattern Sequences as an Indication of Plagiarism," *Journal of Software Engineering and Applications*, 2012, April 2012.
9. Zeidman, Bob, "Will Congress Break the Internet?" *IPWatchdog* (http://www.ipwatchdog.com/2012/02/08/will-congress-break-the-internet/id=22143), February 2, 2012.
10. Zeidman, Bob, "The Case of the Arrogant Expert," *Intellectual Property Today*, February 2012.
11. Zeidman, Bob, "The Software IP Detective: Infringement Detection in a Nutshell," *IPWatchdog* (http://www.ipwatchdog.com/2011/11/20/the-software-ip-detective-infringement-detection-in-a-nutshell/id=20495), November 20, 2011.
12. Zeidman, Bob, "How Do I Infringe Thee? Let Me Count the Ways," *InformIT* (http://www.informit.com/articles/article.aspx?p=1750207&seqNum=3), Sep 21, 2011.
13. Zeidman, Bob, "The history of digital game intellectual property from Atari to Zynga," The Museum of Art and Digital Entertainment, August 4, 2011.
14. Shay, I., Baer, N., and Zeidman, R., "Measuring Whitespace Patterns in Computer Source Code as an Indication of Plagiarism," *Intellectual Property Today*, October 2010.
15. Zeidman, B., "Software v. Software," *IEEE Spectrum*, October 2010.
16. Shay, I., Baer, N., and Zeidman, R., "Measuring Whitespace Patterns as an Indication of Plagiarism," ADFSL Conference on Digital Forensics, Security and Law, May 20, 2010.
17. Hoehn, T. and Zeidman, R., "Measuring the Speedup of a Commercial Application on a Computer Grid," ISCA 22nd International Conference On Parallel And Distributed Computing And Communication Systems, September 24, 2009.
18. Zeidman, R., "DUPE: The Depository of Universal Plagiarism Examples," 5th International Conference on IT Security Incident Management & IT Forensics, September 2009.
19. Baer, N. and Zeidman, B., " Measuring Changes in Software with CLOC," *Embedded.com* (http://www.embedded.com/design/prototyping-and-development/4008330/Measuring-Changes-in-Software-with-CLOC), July 28, 2009.
20. Baer, N. and Zeidman, B., "Measuring Changes in Software IP," *Intellectual Property Today*, June 2009.
21. Baer, N. and Zeidman, B., "Measuring Software Evolution with Changing Lines of Code," 24th International Conference on Computers and Their Applications (CATA-2009), April 10, 2009.
22. Zeidman, B., "Detecting and Proving Software Theft and Infringement," SDForum: Emerging Tech SIG, December 10, 2008.
23. Zeidman, B., "Find Your Copy-Cats With SCC," *Software Test & Performance*, October 2008, pp 18-23.
24. Zeidman, B., "Software Intellectual Property," invited guest lecture, NALSAR University of Law, Hyderabad, India, October 27, 2008.
25. Zeidman, B., "Detecting Stolen Code," EDAC Anti-Piracy Committee, June 12, 2008.
26. Zeidman, R., "Multidimensional Correlation of Software Source Code," The Third International Workshop on Systematic Approaches to Digital Forensic Engineering, May 22, 2008.
27. Zeidman, B., "Getting Better Search Results," *Dr. Dobb's Journal*, May 2008, pp 44-48.

28. Zeidman, B. and Baer, N., "What, Exactly, Is Software Trade Secret Theft?" *Intellectual Property Today*, March 2008.

29. Berger, A., Hill, M., and Zeidman, B., "Software and RTOS synthesis: The next step in software development?" *Programmable Logic Design Line* (http://www.eetimes.com/design/programmable-logic/4015158/Software-and-RTOS-synthesis-The-next-step-in-software-development-), February 27, 2008.

30. Zeidman, R., "Iterative Filtering of Retrieved Information to Increase Relevance," *Journal of Systemics, Cybernetics and Informatics*, Vol. 5 No. 6, 2007, pp 91-96 (reprint).

31. Hoehn, T. and Zeidman, B., "Grid-Enabling Resource Intensive Applications," *Dr. Dobb's Journal*, November 2007, pp 22-28.

32. Zeidman, R., "Iterative Filtering of Retrieved Information to Increase Relevance," The 11th World Multi-Conference on Systemics, Cybernetics and Informatics, July 11, 2007.

33. Zeidman, B., "Who Stole My Software?" High Technology Crime Investigation Association, May 10, 2007.

34. Zeidman, B., "How to choose an RTOS for your FPGA and ASIC designs," *Programmable Logic Design Line* (http://www.eetimes.com/design/programmable-logic/4015109/How-to-choose-an-RTOS-for-your-FPGA-and-ASIC-designs), May 10, 2007.

35. Zeidman, B., "Sharing Videos With Invisible Commercials," The Thirty Third Asilomar Microcomputer Workshop, April 18, 2007.

36. Zeidman, B., "Real Time Operating Systems for Systems on a Chip," IEEE Consultants Network of Silicon Valley, April 17, 2007.

37. Zeidman, B., "What, Exactly, Is Software Plagiarism?" *Intellectual Property Today*, February 2007.

38. Zeidman, B., "Software Synthesis for Embedded Systems," Silicon Valley Code Camp, October 8, 2006.

39. Zeidman, B., "Who Stole My Code," Silicon Valley Code Camp, October 8, 2006.

40. Zeidman, B., "Software Source Code Correlation," 5th IEEE/ACIS International Conference on Computer and Information Science, July 12, 2006.

41. Zeidman, B., "The Death of the Structured ASIC," *Chip Design*, April/May, 2006 (reprint).

42. Zeidman, B., "The Death of the Structured ASIC," *Programmable Logic Design Line* (http://chipdesignmag.com/display.php?articleId=386), April 18, 2006.

43. Zeidman, B., "All about FPGAs," *Programmable Logic Design Line* (http://www.design-reuse.com/articles/12884/all-about-fpgas.html), March 22, 2006.

44. Zeidman, B., "The pluses and minuses of being a consultant," IEEE Gold, Santa Clara Valley Chapter, February 9, 2006.

45. Zeidman, B., "Using software synthesis for multiprocessor OS and software development," *Embedded.com* (http://www.embedded.com/design/prototyping-and-development/4006506/Using-software-synthesis-for-multiprocessor-OS-and-software-development), January 6, 2006.

46. Zeidman, B., "Implementing Integrated Hardware/Software Projects With High Level Languages and Tools," Boston Section of the IEEE, December 1, 2005.

47. Zeidman, B., "Back to the basics: Programmable Systems on a Chip," *Embedded.com* (http://www.eetimes.com/design/programmable-logic/4014776/Introduction-to-Programmable-Systems-on-a-Chip), July 27, 2005.

48. Zeidman, B., "Who Stole My Code?" The Thirty First Asilomar Microcomputer Workshop, April 21, 2005.

49. Zeidman, B., "Software Synthesis for OS-Independent Coding," *Dr. Dobb's Journal*, April 2005, pp 58-63.

50. Zeidman, B., "RTOS Synthesis for Embedded Systems," Server Blade Summit, March 24, 2005.

51. Zeidman, B., "RTOS Synthesis to Reduce Power Consumption," DesignCon 2005, February 1, 2005.

52. Zeidman, B., "Software synthesis for embedded systems," *Embedded Systems Programming*, February 2005, pp 36-43.

53. Zeidman, B., "Testing a Network Device Prototype in a Live Network," *Annual Review of Communications* Volume 4, 2004, p.803 (reprint).

54. Zeidman, B., "Roll Your Own Real-Time OS," *BladeLetter*, Q3 2004, p. 8.

55. Zeidman, B., "Detecting Source-Code Plagiarism," *Developer 2.0*, September 2004 (reprint).

56. Zeidman, B., "Detecting Source-Code Plagiarism," *Dr. Dobb's Journal*, July 2004, pp 55-60.

57. Zeidman, B., "Software Synthesis for Embedded Systems," The Thirtieth Asilomar Microcomputer Workshop, April 27, 2004.

58. Zeidman, B., "Smart Pills: The Royal Treatment," *IP Law & Business*, March 2004.

59. Zeidman, B., "Emulating/Prototyping a Network Device in a Live Network," DesignCon 2004, February 3, 2004.

60. Zeidman, B., "Software synthesis is productive for system design," *EE Times* (online), January 12, 2004.

61. Zeidman, B., "Guidelines for Effective E-Learning," *Chief Learning Officer*, December, 2003, pp. 24-31.

62. Zeidman, B., "Universal Design Methodology," *Embedded Systems Programming*, December, 2003, pp 55-56.

63. Zeidman, B., "Testing a network device in a live network," *EE Times*, October 20, 2003, p.92 (reprint).

64. Zeidman, B., "Test Your Next Hardware Design—in a Live Network," *Communication Systems Design*, October 2003, p.18.

65. Zeidman, B., "Platform FPGAs to Prevail," *Embedded Systems Programming*, November, 2003, pp.28-31.

66. Zeidman, B., "The Universal Design Methodology -- taking hardware from conception through production," *EDN*, December 26, 2002, p53.

67. Zeidman, B., "FPGAs vs. ASICs for Networking," Network Processor Conference West, October 2002, 11pp.

68. Zeidman, B., "How to Start A Consulting Business," *Embedded Systems Programming*, December 2000, pp161-163.

69. Zeidman, B., "An Introduction to Remote Backup," *Disaster Recovery Journal*, Vol. 9 No. 2, April/May/June 1996, p48.

70. Zeidman, B., "Testing System Memory Quickly and Efficiently," Design SuperCon 96 conference, February 1, 1996, 14pp.

71. Zeidman, B., "Remote Backup - Transmitting Critical Data Over Phone Lines for Offsite Storage," JAMCON '95 Communications Conference, August 20, 1995, pp97-99.

72. Zeidman, B., "Testing Memory Quickly," *Embedded Systems Programming*, Aug 1995, pp68-75
73. Zeidman, B., "Read-Ahead Logic: An Alternative to Cache," Design SuperCon 95 conference, March 1, 1995, 6pp.
74. Zeidman, B., "How to Make Money as a Consultant," *Income Opportunities*, Dec 1993, pp48-62
75. Zeidman, B., "Interleaving DRAMs for Faster Access," *ASIC & EDA*, November 1993, pp24-34.
76. Zeidman, B., "Starting Up Your Engineering Consulting Business," *High Technology Careers*, April/May 1993, p24.
77. Zeidman, B., "New Film Software for Independent Producers," *CUE*, September 1992, pp6-7.
78. Hafeman, D. and Zeidman, B., "Memory Architectures Compound RISC's Gains," *Electronic Design*, July 11, 1991, pp71-82.
79. Flynn, M. J., Zeidman, R. and Lochner, E., "Sparse Distributed Memory Prototype: Address Module Hardware Guide," Stanford University Computer Systems Laboratory CSL-TR-88-373, December 1988 72pp.
80. Flynn, M. J., Kanerva, P., Ahanin, B., Bhadkamkar, N., Flaherty, P., Hickey, P., Lochner, E., Webber, K., and Zeidman, R., "Sparse Distributed Memory Prototype: Principles of Operation," Stanford University Computer Systems Laboratory CSL-TR-88-338, December 1988 96pp.

PATENTS

I am a named inventor on following patents and patent applications:

1. Zeidman, Robert M., "Visual tool for developing real time task management code," U.S. Patent 6,934,947.
2. Zeidman, Robert M., "Method for connecting a hardware emulator to a network," U.S. Patent 7,050,962.
3. Zeidman, Robert M., Hafeman, Daniel R., Barr, Michael, "Method and apparatus for synthesizing a hardware system from a software description," U.S. Patent 7,210,116.
4. Zeidman, Robert M., "Apparatus and method for connecting hardware to a circuit simulation," U.S. Patent 7,266,490, RE 42,227.
5. Zeidman, Robert M., "Software tool for detecting plagiarism in computer source code," U.S. Patent 7,503,035.
6. Zeidman, Robert M., Hafeman, Daniel R., Barr, Michael, "Method and apparatus for synthesizing a hardware system from a software description," U.S. Patent 7,620,928.
7. Zeidman, Robert M., Hafeman, Daniel R., "Method and apparatus for emulating a hardware/software system using a computer," U.S. Patent 7,647,583.
8. Zeidman, Robert M., "Detecting plagiarism in computer source code," U.S. Patent 7,823,127.
9. Zeidman, Robert M., "Apparatus and method for connecting hardware to a circuit simulation," U.S. Patent 7,835,897.
10. Zeidman, Robert M., "Software tool for synthesizing a real-time operating system," U.S. Patent 7,882,488.
11. Zeidman, Robert M. and Snider, Gregory, "Using readily available driver and application source code with a synthesized operating system," U.S. Patent 7,900,187.
12. Zeidman, Robert M., Hafeman, Daniel R., Barr, Michael, "Method and apparatus for synthesizing a hardware system from a software description," U.S. Patent 7,945,879.

13. Zeidman, Robert M., "System and method for connecting a logic circuit simulation to a network," U.S. Patent 8,160,863.
14. Zeidman, Robert M. "Use of hardware peripheral devices with software simulations," U.S. Patent 8,195,442
15. Zeidman, Robert M., "Detecting copied computer source code by examining computer object code," U.S. Patent US 8,255,885.
16. Zeidman, Robert M., "Software tool for detecting plagiarism in computer source code," U.S. Patent 8,261,237.
17. Zeidman, Robert M., "Method for advertisers to sponsor broadcasts without commercials," U.S. Patent 8,316,390.
18. Zeidman, Robert M., "Conveying Data From A Hardware Device To A Circuit Simulation," U.S. Patent 8,380,481.
19. Zeidman, Robert, "Software for filtering the results of a software source code comparison," U.S. Patent 8,495,586.
20. Zeidman, Robert, and Hoehn, Timothy, "Searching the internet for common elements in a document in order to detect plagiarism," USPTO application number 12/253,249 (pending).
21. Mylroie, Steve and Zeidman, Robert M., "Detecting plagiarism in computer markup language files," USPTO application number 12/870,817 (pending).
22. Zeidman, Robert M., "Detecting plagiarism in computer source code," USPTO application number 12/871,808 (pending).
23. Zeidman, Robert M., "Method for advertisers to sponsor broadcasts without commercials," USPTO application number 13/632,574 (pending).
24. Zeidman, Robert M., "Conveying Data From A Hardware Device To A Circuit Simulation," USPTO application number 13/766,960 (pending).

TRAINING/TEACHING EXPERIENCE

I have presented seminars and courses on the following topics:

- Analysis of Software Copyright Infringement Cases
- ASIC Design
- CPLD Design
- Detecting Software IP Theft
- Electrical Engineering for non-EEs
- Finding and Utilizing Technical Consultants for IP Litigation
- FPGA Design
- FPGAs vs. ASICs for Networking
- The History of Digital Hardware Design
- How to Start a Consulting Business
- Investigating Technology Theft
- Measuring Software Changes with the CLOC Method
- Memory Architectures
- Patents
- Programmable Systems on a Chip (SOCs)
- Real-Time Operating Systems for SOCs
- Software Intellectual Property
- Software Synthesis
- Starting a Consulting Business
- Technical Consultants for IP Litigation

- Testing Memory
- Universal Design Methodology
- Verilog and HDLs

At the following places:

- Cogswell College
- Cornell Entrepreneur Network
- DesignCon
- Eastcon
- Embedded Systems Conference Boston
- Embedded Systems Conference Chicago
- Embedded Systems Conference Europe
- Embedded Systems Conference India
- Embedded Systems Conference Silicon Valley
- Gigabit Ethernet Conference
- High-Level Electronic System Design Conference
- High Technology Crime Investigation Association
- Institute of Electrical and Electronics Engineers
- Microsoft Store, Stanford Shopping Center
- Network Processors Conference West
- Northcon
- Palo Alto Area Bar Association
- PCB Design East
- PCB Design West
- San Francisco State University
- Semizone.com
- Southcon
- Stanford University
- Westcon
- World Intellectual Property Technical Forum

Exhibit B: Select Cellebrite bootloader exploit upload addresses

These addresses were obtained from the file

`..\Genesis 1190RC 9JAN2012\Genesis\XML DB\DataFiles\Phones\__Dump_SamsungGSM.xml.`

SGH-Z560V	0141A000
SGH-Z400	01800000
SGH-F480T	01800000
GT-S5600 Player Star	01800000
GT-S5603	01800000
SGH-Z170	02510000
SGH-Z230	02510000
SGH-Z240	02510000
SGH-G400	02510000
SGH-A411	02510000
SGH-A597 Eternity 2	02510000
SGH-T639	02510000
SGH-T659	02510000
SGH-F700v	02510000
SGH-A707 SYNC	02510000
SGH-A727	02510000
SGH-A737	02510000
SGH-L810V	02510000
SGH-A887 Solstice	02510000
SGH-U900	02510000
SGH-U908	02510000
SGH-T929	02510000
GT-S3370B	02510000
GT-S5350	02510000
GT-S5510T	02510000
SGH-M7500 Armani	02510000
SGH-M7600	02510000
SGH-M7603	02510000
GT-S7220	02510000
SGH-S7350	02510000
SGH-S8300	02510000
SGH-M8800 Pixon	02510000
SGH-F480	0252F000
SGH-T469 Gravity 2	02800000

SGH-T746	02800000
SGH-T749 Highlight	02800000
SGH-A847 Rugby II	02800000
SGH-A847R	02800000
SGH-A867	02800000
SGH-A877	02800000
SGH-A886 Forever	02800000
SGH-A897 Mythic	02800000
SGH-T919 Behold	02800000
SGH-A927	02800000
SGH-U800	01800000
SGH-ZV60	02510000
SGH-L170	02510000
SGH-F330	02510000
SGH-F400	02510000
SGH-F490	02510000
SGH-F490V	02510000
SGH-A551	02510000
SGH-A561	02510000
SGH-J630	02510000
SGH-A637	02510000
SGH-A717	02510000
SGH-Z720	02510000
SGH-A747 SLM	02510000
SGH-L760	02510000
SGH-A766 Propel	02510000
SGH-A767 Propel	02510000
SGH-L770	02510000
SGH-A777	02510000
SGH-G800	02510000
SGH-G808	02510000
SGH-T819	02510000
SGH-A827	02510000
SGH-A837 Rugby	02510000
GT-C5220	02510000
SGH-S7330	02510000
SGH-A711	02800000
SGH-A736	02800000

Exhibit C: Summary of diagnostic packets from Samsung serial data interface document

Packet Name	Code	DM Can Send	Mode	Supported
DMSS Version Number	0	Yes	Any	Yes
DMSS ESN	1	Yes	Any	Yes
Memory Peek Byte	2	Yes	Any	Yes
Memory Peek Word	3	Yes	Any	Yes
Memory Peek Dword	4	Yes	Any	Yes
Memory Poke Byte	5	Yes	Any	Yes
Memory Poke Word	6	Yes	Any	Yes
Memory Poke Dword	7	Yes	Any	Yes
Port Output Byte	8	Yes	Any	Yes
Port Output Word	9	Yes	Any	Yes
Port Input Byte	10	Yes	Any	Yes
Port Input Word	11	Yes	Any	Yes
Status	12	Yes	Any	Yes
Vocoder Memory Peek	13	Yes	Any	Yes
Vocoder Memory Poke	14	Yes	Any	Yes
Logging Mask	15	Yes	Any	Yes
Log Request	16	Yes	Any	Yes
Non-Volatile Memory Peek	17	Yes	Any	Yes
Non-Volatile Memory Poke	18	Yes	Any	Yes
Bad Command Response	19	No	Any	Yes
Bad Parameters Response	20	No	Any	Yes
Bad Length Response	21	No	Any	Yes
Invalid Device Response	22	No	Any	No
Vocoder Error Response	23	No	Any	Yes
Bad Mode Response	24	No	Any	Yes
Temporal Analyzer Graph	25	Yes	Any	Yes
Markov Statistics	26	Yes	Any	Yes
Markov Statistics Reset	27	Yes	Any	Yes
Diag Version	28	Yes	Any	Yes

Time Stamp	29	Yes	Any	Yes
Temporal Analyzer Parameter	30	Yes	Any	Yes
DMSS Message	31	Yes	Any	Yes
Handset Emulation Keypress	32	Yes	Any	Yes
Handset Emulation Lock	33	Yes	Any	Yes
Parameter Retrieval	35	Yes	Any	Yes
Parameter Set	36	Yes	Any	Yes
Non-Volatile Item Read	38	Yes	Any	Yes
Non-Volatile Item Write	39	Yes	Offline	Yes
Table Configuration	40	Yes	Any	No
Mode Change	41	Yes	Any	Yes
Error Record Retrieval	42	Yes	Any	No
Error Record Clear	43	Yes	Any	No
Symbol Error Rate Reset	44	Yes	Any	Yes
Symbol Error Rate Report	45	Yes	Any	Yes
DIP Switch Retrieval	47	Yes	Any	Yes
DIP Switch Set	48	Yes	Any	Yes
Vocoder PCM Loopback	49	Yes	Any	Yes
Vocoder PKT Loopback	50	Yes	Any	Yes
Data TX	51	Yes	Any	No
Data RX	52	Yes	Any	No
Call Origination	53	Yes	Any	No(TBD)
Call End	54	Yes	Any	Yes
Local Service Option	55	Yes	Any	No
Main Control State	56	Yes	Any	No
Call Answer	57	Yes	Any	No
Switch to Downloader	58	Yes	Offline	No
Sequence Numbers	60	Yes	Any	Yes
Sleep	61	Yes	Any	Yes
System Time	62	Yes	Any	Internal Use
Unused	63	No		No
Phone State	64	Yes	Any	No(TBD)
Pilot Sets	65	Yes	Any	Yes

Service Programming Code	66	Yes	Any	Yes
Bad SPC Mode Response	67	No	Any	Yes
Parm Get 2	68	Yes	Any	Yes
Reset NV	69	Yes	Any	Yes

Exhibit D: BitMatch results

TO BE PRODUCED IN CD-ROM FORMAT